3. Clases de funciones

Una concluisión que hemos obtenido en la sección anterior es que el mejor algoritmo es aquél que resuelve el problema planteado de acuerdo con la situación. La respuesta para el caso general, para todas las situaciones, la hemos dado generalizando, también, el tamaño del ejemplar. Es por ello que hablamos del comportamiento asintótico de un algoritmo. Vamos a ver en esta sección que es posible manipular los comportamientos de los algoritmos para llevar a cabo análisis más detallados de ellos. Justificaremos, por ejemplo, el uso de la "escala de funciones" respecto de su comportamiento (contante, logarítmica, lineal, cuadrática, cúbica etc.). Particularmente, nos interesa saber qué tanto conviene mejorar un algoritmo si cambiamos una parte él. Aunque esta pregunta está relacionada además con el diseño, requeriremos de una serie de reglas para determinar con rapidez cuestiones como la anterior.

3.1. Cota superior

Formalizaremos en primer lugar la noción usada con anterioridad para referirnos a que un algoritmo es mejor que otro o bien que un algortitmo a lo más se comporta tan lento como otro. Por supuesto, haremos estas convenciones retomando la idea de que un algoritmo se caracteriza por una *función de comportamiento* dependiente del tamaño del ejemplar cuyo valor es el tiempo que requiere el algoritmo para realizar su tarea. Conviene, asimismo, señalar que todas estas funciones son no negativas pues siempre un algoritmo tarda al menos cero unidades, lo cual supondremos sin indicar. Desde que se planteó la necesidad de comparar funciones de comportamiento, digamos f y g, fue mencionado que para afirmar que el algoritmo de g no podía ser más lento que el de f, debía cumplirse que g no rebasara a un múltiplo de f para ejemplares de tamaño mayores o iguales que N. Esto significa, que un algoritmo, con función g, será mejor o a lo más igual que otro, con función f, si $g(n) \leq cf(n)$ para toda $n > N$ y cierta constante $c > 0$ (recordemos que el principio de invarianza refería a dos desigualdades para indicar el mismo comportamiento). Precisando: Se dice que una función $g : \mathbb{N} \to \mathbb{R}$ está en *el orden de* $f : \mathbb{N} \to \mathbb{R}$ sii existe $c > 0$ tal que $g(n) \leq cf(n)$ para toda $n \geq N$, $N \in \mathbb{N}$. Destacamos que para afirmar que g está en el orden de f debe determinarse c, el múltiplo, y N, el umbral, de tal forma que $g(n) \leq cf(n)$ para toda $n \geq N$. Notemos que dada una función f, habrá muchas funciones g que satisfagan esta relación, por ejemplo si $f(n) = n^2$, entonces están en el orden de f: $g_1(n) = n$, $g_2(n) = \log(n)$, $g_3(n) = n\log(n)$, $g_4(n) = 2n^2$, etc. Llamaremos a esta clase de funciones $O(f(n))$. Así:

$$O(f(n)) = \{g : \mathbb{N} \to \mathbb{R} | (\exists c > 0, N \in \mathbb{N})(\forall n \geq N)(g(n) \leq cf(n))\}.$$

$O(f(n))$ contiene a todas las funciones que están *acotadas superiormente* por un múltiplo de f.
La definición de g "está en el orden de" f contiene una redundancia al exigir la determinación tanto del múltiplo como del umbral. La constante c permite que el múltiplo de f supere a una función g siempre que ésta no crezca más rápido que f. Pero también es posible que en un intervalo finito la función f tome valores negativos, lo cual se evita "saltando" este intervalo al dar un valor de umbral mayor que el límite superior de ese intervalo. Lo mismo podemos hacer cuando f toma valor cero. Ya que las funciones de comportamiento no toman valores negativos, aunque sí el valor cero, podemos llevar nuestra observación hacia una formulación más simple de la clase de funciones con cota superior f. Lo anterior es una justificación de la proposición 8.

Proposición 8 (Regla del Umbral) Sea f estrictamente positiva. $g(n) \in O(f(n))$ si y sólo si existe $c > 0$ tal que $g(n) \leq cf(n)$, para toda $n \in \mathbb{N}$.

Al afirmar que una función g está en la clase $O(f(n))$ no sólo queremos decir que el algoritmo de g no puede ser más lento que el de f, además la función f puede ser empleada como un "punto en la escala de funciones" y de este modo ubicar a g en relación con los demás algoritmos en la escala. Las funciones de comportamiento son expresiones obtenidas del análisis de los algoritmos y será común encontrar que un algoritmo esté compuesto de más de un paso, lo cual se traducirá en que su función de comportamiento sea la suma correspondiente de las funciones de sendos pasos. Con el fin de determinar la ubicación de una función en la escala, y de manera semejante al razonamiento que se hace con los límites, cuando f es la suma de funciones $\{f_i\}_i$ podremos despreciar algunas funciones para quedarnos con la función representativa de f. El hecho implícito en la idea expuesta es que habrá en está suma alguna función f_j que no pueda ser superada por las demás, entonces $O(\sum_i f_i(n)) = O(f_j)$. Por ejemplo, $O(n^2 + n) = O(n^2)$, esto es, $\max\{n^2, n\} = n^2$. Usamos el operador $\max\{f_j(n), f_i(n)\} = f_j(n)$ para indicar que f_j siempre supera a f_i excepto en un número finito de puntos: $f_i(n) < f_j(n)$, para toda $n > N$. Este resultado es útil y por ello lo enunciamos como la *Regla del Máximo*.

Proposición 9 (Regla del Máximo) Si f_1 y f_2 son funciones de comportamiento, entonces $O(f_1(n) + f_2(n)) = O(\max\{f_1(n), f_2(n)\})$.

Debe cuidarse la premisa implícita establecida al inicio: las funciones de comportamiento son no negativas, y, en muchos casos, estrictamente positivas. Un ejemplo que muestra la obtención de errores al violar la premisa antedicha es: $O(n) = O(n + n^2 - n^2) = O(\max\{n, n^2, -n^2\}) = O(n^2)$. El error es que aparece una función que no es de comportamiento (positiva).

Observación 5

1. En el comentario del primer párrafo tenemos claramente que $g_i(n) \in O(f(n))$, $i = 1, 2, 3, 4$. Pero además, si $h(n) = n^3$, entonces $f(n) \in O(h(n))$, puesto que con $c = 1$ y $N = 1$ se cumple que $n^2 \leq cn^3$ para toda $n \geq N$. Por otro lado, es fácil explicar por qué también $g_i(n) \in O(h(n))$: si sabemos que $g_i(n) \leq c_i f(n)$ y que $f(n) \leq ch(n)$, entonces $g_i(n) \leq cc_i h(n)$, esto es $g_i(n) \in O(h(n))$. Lo anterior puede sostenerse sin que haya referencia alguna a los ejemplos particulares que hemos dado, bastan las definiciones e hipótesis.
2. Un hecho importante pero aparentemente sin utilidad es que para toda función de comportamiento f se cumple que $f(n) \in O(f(n))$.
3. Es posible aún imaginar que un par de funciones f, g cumpla $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$. Aquí también es fácil ver que $O(f(n)) = O(g(n))$, pues para cualquier f_1 tal que $f_1 \in O(f(n))$ se tiene $f_1(n) \leq cf(n)$ $(n > N_1)$, pero además por hipóteis $f(n) \leq dg(n)$ $(n > N_2)$ luego para toda $f_1 \in O(f(n))$: $f_1(n) \leq cdg(n)$ $(n > \max N_1, N_2)$. Por tanto $O(f(n)) \subset O(g(n))$. Similarmente puede verse que $O(g(n)) \subset O(f(n))$.
4. Si convenimos definir la relación "$\in O$", ésta es como vimos reflexiva, transitiva y antisimétrica, y por tanto es una relación de orden.

Ejercicio 18

1. Demuestre o refute las siguientes afirmaciones:
   a) $n^3 \in O(n^2)$.
   b) $2^{n+1} \in O(2^n)$.
   c) $n! \in O((n+1)!)$.
2. Sea $k \in \mathbb{N}$. Demuestre que si una función de comportamiento f expresada por $f(n) = f_1(n) + f_2(n) + \cdots + f_k(n)$ donde $f_1, f_2, \ldots, f_k$ son también funciones de comportamiento, y $g(n) = \max\{f_1(n), f_2(n), \ldots, f_k(n)\}$ entonces $O(f(n)) = O(g(n))$.
3. Usando el resultado del problema anterior, por qué no es posible afirmar que

$$O(n) = O(\max\{\overbrace{n, \ldots, n}^{n \text{ veces}}\}) = O(\overbrace{n + \cdots + n}^{n \text{ veces}}) = O(n^2).$$