

Complejidad

La complejidad computacional es la clasificación de los problemas en función del tamaño de su entrada. En particular, nos interesa el análisis de cómo se comporta un algoritmo cuando el tamaño de la entrada (input) es lo suficientemente grande.

¿Por qué Big-O?

Si bien hay 3 cotas: Omega (Ω) o el mejor caso, Theta (Θ) o el caso promedio y Big-O (O) o el peor caso. Para este ramo nos centraremos casi completamente en el Big-O, el peor caso.

¿Por qué nos interesa el peor caso más que el mejor de los casos? Un buen ejemplo del porqué nos interesa el peor caso puede ser el peso que soporta un ascensor, en el caso que dijéramos “Este ascensor puede levantar más de 10kg”, una pregunta razonable es qué tanto más que 10kg puede levantar, ¿puede levantar 10 personas? ¿una carga de toneladas de metal? No lo sabemos, solo tenemos la información que la acota inferiormente. Un dato más importante podría ser la capacidad de carga máxima, es decir, saber que “Este ascensor no puede levantar más de 500kgs” nos habla de qué casos podría servirnos y cuando no.

Principales ordenes de complejidad

Si bien, los ordenes de complejidad pueden ser cualquier función, los órdenes más comunes son:

$O(1)$	constante	Mostrar un dato en pantalla; acceder a un elemento en un arreglo
$O(\lg(N))$	logarítmica	Busqueda binaria; Recorrer una rama en un árbol
$O(N)$	lineal	Busqueda lineal, Suma
$O(N^2)$	cuadrática	BubbleSort; Multiplicación (cuadrática cuando $N = M$, $O(M*N)$)
$O(N^3)$	cúbica	Multiplicación de matrices
$O(2^N)$	Exponencial	Conjunto potencia
$O(N!)$	Factorial	Recorrer todos los caminos posibles de un grafo

Búsqueda lineal (Un ejemplo de una complejidad lineal)

La búsqueda lineal es la búsqueda sobre un conjunto iterando uno por uno, en cualquier orden (de izquierda a derecha o de derecha a izquierda). Esta búsqueda tiene un peor caso lineal. Por ejemplo, si tuviéramos un conjunto de 100 números y quisiéramos saber si el número 58 está incluido dentro de ese conjunto, tendríamos que buscar uno por uno dentro de los 100 números para ver si alguno de esos es el 58. Es decir, en el peor de los casos revisaremos 100 números. En general, diremos que si tenemos N números y queremos revisar si un número existe dentro de ese conjunto, tendremos un Big-O de $O(N)$.

Busqueda binaria (Un ejemplo de una complejidad logarítmica)

La búsqueda binaria es la búsqueda sobre un conjunto **ordenado**, en el que usaremos la transitividad (si $a > b$ y $b > c$, entonces $a > c$) para poder descartar casos de una manera más eficiente.

Tengamos el siguiente arreglo: [5, 6, 8, 9, 15, 37, 73, 93, 144] y queremos saber si existe o no el 58 dentro de este arreglo. El algoritmo consiste en usar el largo del arreglo (en este caso 9) para partirlo por la mitad (usando números enteros, usemos el 4 en este caso) y preguntarse si el elemento en ese índice es o no 58, luego revisar si es menor o mayor. En este caso, nos pararemos sobre el elemento 9 y nos daremos cuenta que es menor que 58, por transitividad entonces, sabemos que todos los elementos menores a 9 también son menores a 58 y solo tendremos que buscar por la derecha.

[5, 6, 8, 9, 15, 37, 73, 93, 144]

Los números en rojo ya no vale la pena revisarlos porque sabemos que todos son menores a 58 solo con saber que 9 es menor que 58. Entonces solo tenemos que volver a revisar en los números en verde, los números [15,

37, 73, 93, 144]. Volvemos a repetir el proceso y nos paramos en la mitad del arreglo, el elemento 73. Como sabemos que el 73 es mayor que 58 podemos descartar también todos los elementos mayores a 73.

[15, 37, 73, 93, 144]

Y comparando contra 2 números el 58, ya descartamos casi todos los elementos. Esto podríamos repetirlo hasta encontrar el elemento o hasta que no queden más elementos por revisar.

La diferencia entre una búsqueda lineal y una búsqueda binaria es gigante, tomemos por ejemplo la diferencia si buscáramos en un conjunto con 100.000.000 elementos. Si hiciéramos una búsqueda lineal tendríamos que hacer más o menos 100.000.000 de operaciones, pero si hicieramos una búsqueda binaria para el mismo trabajo tendríamos que hacer del orden de las 27 operaciones.

Video de una búsqueda binaria bailando flamenco.

Suma (o por qué el marco de referencia importa)

La Suma generalmente la consideramos una operación constante. Por ejemplo, si tuviéramos que sumar 1.000.000 de números, un análisis podría decir que nos tomará más o menos 1.000.000 de sumas. Es decir, esta operación es $O(N)$.

Pero asumir que la suma es constante no siempre es lo correcto, si recordamos cómo aprendimos a sumar en el colegio, tenemos la intuición de que entre más dígitos tengan los números a sumar, mayor serán las operaciones que tendremos que hacer para llegar a un resultado.

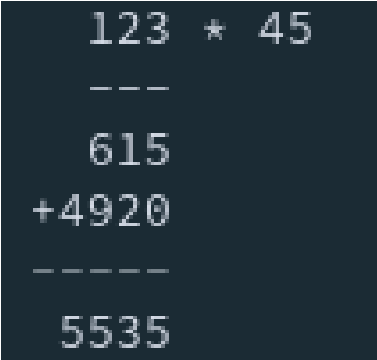
Esto nos dice que sumar 2 números, nos tomará $O(D)$ donde D es la cantidad de dígitos que tiene el mayor número. Entonces en el problema de sumar 1.000.000 de números probablemente tengamos un análisis más exacto diciendo que $O(N \cdot D)$ donde N es la cantidad de números a sumar y D es la cantidad de dígitos que tiene el número con más dígitos de todos los N números.

Pero la pillería está en que D , si estamos usando el sistema decimal tendrá un valor de $\log_{10}(X)$ donde X es el número que queremos conocer la cantidad de dígitos que tiene. Es por esto que D , salvo casos excepcionales donde se trabaja con números demasiado grandes, podría considerarse un número lo suficientemente pequeño como para considerarse constante.

Finalmente, ambos análisis están correctos con el marco de referencia necesario. El análisis de complejidad no es solamente una actividad mecánica.

Multiplicación

Multiplicar 2 números, por ejemplo el 123 multiplicado por 45, nos tomará una cantidad de operaciones elementales relacionada a los dígitos que tienen ambos números. La multiplicación que aprendimos en el colegio comienza multiplicando 123 por 5, lo que nos dará 615 y luego, multiplicar 123 por 4 que nos dará 492 y esto lo multiplicaremos por 10 para mantener la posición que corresponde, es decir, 4920.



```
123 * 45
 ---
 615
+4920
 ----
 5535
```

Intuitivamente, podríamos decir que la multiplicación tiene una complejidad de $O(N \cdot M)$ donde N y M son la cantidad de dígitos que tiene cada número. En el caso particular donde $N = M$, tendremos que la complejidad es $O(N^2)$. Para ver las implicancias de que la multiplicación sea cuadrática, si tuvieramos que sumar 2 números donde ambos tienen 1000 dígitos, tendremos más o menos 1000 operaciones. Si tuvieramos que multiplicar 2

números donde ambos tienen 1000 dígitos, tendríamos que hacer aproximadamente 1.000.000 operaciones. Gran diferencia.

Hasta hace no mucho (en la década de los 50), Kolmogorov, matemático ruso y uno de los principales científicos en la teoría de la complejidad algorítmica postulaba que la multiplicación tiene un mejor caso de $\Omega(N^2)$. No fue sino hasta inicios de los 60 en una conferencia donde propone esta cota y un estudiante que asistió a esta conferencia luego de escuchar esto encontró un algoritmo en donde su peor caso era menor a N^2 . El cómo lo hizo se verá mucho mejor cuando aprendamos la técnica de dividir y conquistar. Aún así, hay un video de 1 minuto que explica cómo funciona la idea detrás del algoritmo de Karatsuba [Video]

Ejercicios propuestos

- ¿Qué complejidad tiene dividir dos números con respecto a la cantidad de dígitos que tienen?
- Implemente una búsqueda lineal y una búsqueda binaria de manera iterativa y de manera recursiva. Mida experimentalmente cuántas comparaciones realiza cada una.