# PROCEDIMIENTO PARA CALCULAR LA COMPLEJIDAD DE UN ALGORITMO RECURSIVO TIPO DIVIDE Y VENCERÁS

## 1. Introducción

Un algoritmo tipo divide y vencerás es un algoritmo recursivo que recibe un problema de un cierto tamaño, digamos *n*, y para resolver dicho problema lo que hace es dividirlo en problemas de menor tamaño, por ejemplo, *n/2*, hasta que el tamaño del problema sea tan pequeño que su resolución resulte trivial. Una vez resuelto los subproblemas de menor tamaño, las soluciones de cada uno de ellos se combinan o fusionan para resolver el problema de tamaño *n*.

Por ejemplo, para buscar el máximo de un vector de tamaño *n*, podemos dividir dicho vector en dos subvectores de tamaño *n/2*. El máximo del vector de tamaño *n* será el máximo de los máximos de los subvectores de tamaño *n/2*. Si el vector tuviese tamaño 1, el problema sería trivial y no haría falta subdividir el vector. Por tanto, el problema de tamaño *n* se resuelve recursivamente mediante la resolución de dos problemas de tamaño *n/2* y la combinación de sus resultados.

Estos algoritmos recursivos tienen asociada una ecuación de recurrencia tipo, cuya resolución es lo que se conoce como el *Teorema Maestro*. La forma de dicha ecuación de recurrencia, así como el teorema maestro aparecen en la transparencia 21 del Tema 2 de la asignatura. La demostración de dicho teorema puede encontrarse en prácticamente cualquier libro serio sobre algoritmia. En concreto, se recomienda, en caso de estar interesado, leer la demostración de Peña (2005).

A continuación se explica cómo interpretar dicho teorema maestro.

## 2. Aplicando el teorema maestro

Aplicar el teorema maestro implica identificar diversos parámetros de un algoritmo recursivo tipo divide y vencerás. Dichos parámetros se muestran en la Tabla 1.

| Parámetro | Significado |
|---|---|
| a | Número de llamadas recursivas que se realizan en el caso recursivo |
| b | Factor por el cual se divide el tamaño del problema en cada llamada recursiva |
| k | Mayor exponente de las polinomios asociados a las complejidades del caso base y de la parte no recursiva de la rama recursiva del algoritmo, asumiendo que en ambos casos se trata de complejidades polinómicas |

Tabla 1. Parámetros a calcular para aplicar el teorema maestro

Por tanto, el procedimiento para aplicar el teorema maestro es tal como sigue:

(1) Identificar cuáles son las ramas recursivas del algoritmo y cuáles corresponden al caso base.
(2) Identificar el número de llamadas recursivas (parámetro *a*) que se realizan en el caso recursivo. Por ejemplo, en el caso del máximo de un vector, planteado en la introducción de este documento, el número de llamadas recursivas sería 2.
(3) Identificar la razón (parámetro *b*) por la cual se divide el tamaño del problema en cada llamada recursiva. Por ejemplo, en el caso del máximo de un vector, planteado en la introducción de este documento, dicha razón sería 2, dado que el problema se reduce a la mitad.
(4) Calcular la complejidad del caso base, como si se tratase un algoritmo iterativo normal. La complejidad resultante, para aplicar el teorema maestro, debe ser polinómica. Sea *k'* el grado de dicho polinomio[1].
(5) Calcular la complejidad de todas aquellas instrucciones que no sean llamadas recursivas dentro del caso recursivo, tal como si fuese un algoritmo iterativo normal. Normalmente estas instrucciones se corresponden con el código necesario para preparar las llamadas recursivas y para combinar los resultados devueltos por las llamadas recursivas. La complejidad resultante, para aplicar el teorema maestro, debe ser polinómica. Sea *k''* el grado de dicho polinomio.
(6) El teorema maestro asume que las complejidades del caso base y de la parte iterativa del caso recursivo son polinómicas y poseen el mismo grado. Por tanto, tomaremos como dicho grado al mayor de los grados de los polinomios asociados a las complejidades del caso base y de la parte iterativa del caso recursivo. Es decir, *k = max(k',k'')*. Esto es posible gracias a que un algoritmo de complejidad $O(n^a)$ se puede siempre convertir en otro equivalente de complejidad $O(n^b)$, siempre que $b > a$. Pensad que bastaría con anidar el algoritmo de complejidad $O(n^a)$ en tantos bucles sin sentido como hiciese falta para incrementarle la complejidad hasta que llegase a *b.*
(7) A continuación se calcula la relación entre a y $b^k$ y se aplica el teorema maestro, sustituyendo los valores de *k* y *a* tal como corresponda.

Ricardo Peña. "*Diseño de Programas: Formalismo y Abstracción*". Pearson Education, 3ª Ed. (2005).

*Pablo Sánchez Barreiro.*

[1] Si el algoritmo tiene complejidad constante, es decir, O(1), k=0, ya que $n^0 = 1$.