RECURRENCIA

Se tiene la siguiente recurrencia: $T(N) \begin{cases} 1 & n = 0 \\ 9T(n-3) & n \geq 1 \end{cases}$

Es de interés analizar que sucede con el T(N) cuando n toma valores mayores o iguales a 1. Siendo esta una recurrencia, se analiza paso a paso la recursión, para determinar, **el termino general.**

Paso$_1$:

$$9T(n-3)$$

Se deja tal cual esta, sin modificarlo. Representa la primera ejecución.

Paso $_2$:

Ahora se debe hacer un reemplazo de n, por el nuevo argumento que recibirá T(N).

$$9[9T(n-3)] - 3$$

En este paso, n-3 reemplaza a n en la definición original, siendo el nuevo argumento. Este es el segundo paso, en el cual nos hemos metido un nivel mas dentro de la recursión. Si simplificamos un poco la expresión, esta queda de la siguiente forma:

$$9^2T(n-6)$$

Paso$_3$

Nuevamente se vuelve a reemplazar, utilizando el argumento obtenido en el paso anterior:

$$9^2[9T(n-6)] - 3$$
$$9^3T(n-9)$$

Luego de ver sucesivos pasos, estamos en condiciones de obtener el paso general, el paso k.

Paso$_K$

En los pasos anteriores, pudimos observar cierto patrón en la formula:
- En cada paso, la formula entera es multiplicada por 9
- En cada paso, el termino n es restado en tres unidades

Estos patrones, son generalizables:

$$9^kT(n-3k)$$

La recursión continua hasta que n= 0. Para continuar, debemos saber que valor tomara k cuando n=0. Debemos entonces, despejar k

$$n - 3k = 0$$
$$-3k = -n$$
$$k = \frac{-n}{-3}$$
$$k = \frac{1}{3}n$$

Ahora procedemos a reemplazar este valor en la función con el termino general:

$$T(n) = 9^{\frac{1}{3}n} T\left(n - 3\left[\frac{1}{3}n\right]\right)$$

Este es el primer resultado de reemplazar k, en el k-ésimo termino. Se pueden hacer algunas simplificaciones:

$$T(n) = 9^{\frac{n}{3}} T(n - n)$$

Luego, n - n = 0, tendremos T(0) que esta dentro de la definición de la recurrencia. Lo reemplazamos por el valor escrito allí (1)

$$T(n) = 9^{\frac{n}{3}} T(0)$$

$$T(n) = 9^{\frac{n}{3}}$$

Finalmente, concluimos que el tiempo de ejecución de esta recurrencia será de $9^{\frac{n}{3}}$

O(N) POR DEFINICIÓN

Antes de empezar hay que recordar la definición de O(N)

T(N) es de O(f(n)) si existen constantes $c > 0$ y n_0 tal que:

$$T(N) \leq c\ f(n)\ \ \forall\ n \geq n_0$$

Debemos hallar un c y un función para que se cumpla en la siguiente expresión:

$$T(n) = 1000\ n^2 + 5n^3 + 4n^2 \log_2 n$$

Se debe separar en términos la función, ademas es necesario determinar el termino de mayor crecimiento. En este caso, n^3 del segundo termino. La separación en términos simplifica el análisis de toda la expresión. Si logramos probar que para cada termino existe una cota, entonces podremos demostrar que la expresión entera tiene una cota.

T_1	$1000\ n^2$	$\leq c_1\ n^3$	$\forall\ n \geq n_0$
T_2	$5n^3$	$\leq c_3\ n^3$	$\forall\ n \geq n_0$
T_3	$4n^2 log_2 n.$	$\leq c_2\ n^3$	$\forall\ n \geq n_0$
T(N)	$1000\ n^2 + 5n^3 + 4n^2 \log_2 n$	$\leq (c_1 + c_2 + c_3)\ n^3.$	$\forall\ n \geq n_0$

Debemos hallar las constantes $c_1\ c_2\ c_3$. Por simplicidad, se pueden tomar los mismos valores de los términos independientes de cada división que hemos hecho. Podemos comprobar que al reemplazar, se mantienen las proposiciones hechas:

T_1	$1000\ n^2$	$\leq 1000\ n^3$	$\forall\ n \geq 1000$
T_2	$5n^3$	$\leq 5\ n^3$	$\forall\ n \geq 5$
T_3	$4n^2 log_2 n.$	$\leq 5\ n^3$	$\forall\ n \geq 5$
T(N)	$1000\ n^2 + 5n^3 + 4n^2 \log_2 n$	$\leq (1000 + 5 + 5)\ n^3.$ $1010\ n^3$	$\forall\ n \geq 1000$

Al hallar un c y una función que satisfaga la expresión dicha en la definición. Podemos decir que

T(N) = 1000 n2 + 5n3 + 4n2 log2n es $O(n^3)$

PREGUNTAS DE TIEMPO DE EJECUCIÓN

Considere un algoritmo A que se ejecuta sobre una computadora que procesa 10.000 operaciones cada décima de segundo. El algoritmo A tiene una función de tiempo de ejecución $T(n) = nlog_{10}n$ y debe resolver un problema de tamaño n = 100.000 ¿Cuanto tiempo va a tardar?

Del enunciado sabemos que:

Procesa información a un ritmo de 10.000 operaciones por décima de segundo

El algoritmo A tiene un tiempo de ejecución de $T(n) = n \log_{10} n$

El problema a resolver tiene tamaño n= 100.000

Con regla de tres simple, podemos averiguar cuanto tiempo tardará. Tenemos que:

10.000 operaciones ..... Una décima de segundo

100.000 operaciones ..... ?

Pero hay un dato que nos esta faltando utilizar. La computadora que tiene un ritmo de 10000 operaciones por décima de segundo, utiliza un algoritmo de tiempo de ejecución con un tiempo de $T(n) = n \log_{10} n$. Esto afectara al n operaciones a realizar. El coste se vera afectado.

$$T(100.000) = 100.000 log_{10} 100.000$$

$$T(100.000) = 100.000 \frac{log_2 100.000}{log_2 10}$$

$$T(100.000) = 100.000 * \frac{5}{1}$$

$$T(100.000) = 500.000$$

Volvemos a plantear la regla de tres simple:

10.000 operaciones 0,1

500.000 operaciones .....?

$\frac{500.000 * 0,1}{10.000} = \frac{50.000}{10.000} = 5$ segundos toma procesar un problema de tamaño n = 100.000