# ARITMETICA DE LA NOTACION O

Mejor conocido como Big O es una notación Asintótica.

Limite Superior: Big O representa el comportamiento en el peor de los casos
Maximo.

Se usa para analizar nuestros algoritmos

## ALGORITMOS RECURSIVOS

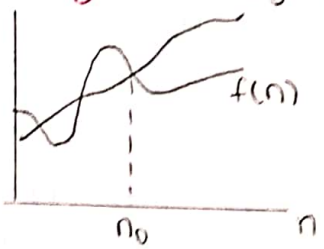
## 3 TECNICAS DIFERENTES.

La notación Big O nos permite fijar una relación en el peor de los casos sobre como va creciendo el tiempo de ejecución de un algoritmo conforme la entrada va incrementando

Definición formal

Una función f(n) pertenece a Big O de otra función g(n) cuando existe una constante positiva c tal que a partir de un valor inicial de n, f(n) no sobrepase a c multiplicando a g(n)

$f(n) = O(g(n))$




$f(n) \leq c \cdot g(n)$

1. Metodo de Sustitución
2. Árbol Recursivo
3. Metodo Maestro

En la normal analizamos su complejidad siempre y cuando el algoritmo no tenga ninguna llamada recursiva Cuando tiene llamadas recursivas hacemos otro tipo de analisis como

Estos son metodos para resolver recurrencias
o que es lo mismo es para determinar la complejidad Big O de recurrencias y llamamos recurrencias a una ecuación que define una función en terminos de esa misma función pero con valores mas pequeños osea es una forma de representar una secuencia recursiva utilizando una formula
Estas recurrencias van muy de la mano con nuestras funciones recursivas porque nos permiten representar los tiempos de ejecución el cuanto tardan esas funciones de una manera matematica

## 1. METODO DE SUSTITUCIÓN

Para este metodo utilizamos inducción matematica para demostrar que la solución funciona (dale una oportunidad no es tan complicado)

Funciona basicamente en dos partes estas son

1. Adivinar → Adivinamos la solución
2. Inducción. → luego utilizamos la inducción matematica para demostrar que esa solución si es la correcta

*Lo dificil es adivinar correctamente la solución

## METODO MAESTRO → Se deriva principalmente del metodo del arbol de recurrencia.

Es simpleniente una serie de reglas y funciones en donde si nuestra función de recurrencia tiene la forma:

$$T(n) = a \cdot T(n/b) + f(n)$$

donde $a \geq 1$ $b > 1$

Nuestra complejidad cae en uno de estos 3 casos.

Lo que hacemos sera tomar nuestra función de recurrencia e identificamos los difertes terminos que vamos a necesitar $a$, $b$, y $f(n)$

La función de recurrencia a fuerza debe ser de la misma forma

1. Tomamos los valores de b y a y calculamos $\log_b(a)$. para luego revisar cual de los 3 casos cae para que mi función $f(n)$ sea igual a Big O de $n^c$ $[f(n) = O(n^c)]$
2. Lo imortante es ver que valor tiene c

   Para ello calculamos $c = \log_b(a) = 1$ si c es exactamente igual para que esto se cumpla entonces cae en el segundo caso
3. Dependiendo de en cual caso caiga reemplazamos los valores que extragimos de nuestra función de recurrencia con los que esta mostrados aquí.
4. y con eso demostramos ya cual es nuestra complejidad.