1. **Aritmética de notación O.**

Decimos que un algoritmo tiene un orden sí existen un y un siendo tal que para todo , La relación denota una dominancia de funciones, en donde la función está acotada superiormente por un múltiplo de la función ( es dominada por ) Así, la expresión refleja que el orden de crecimiento asintótico de la función es inferior o igual al de la función . (Diaz, N., 2014).

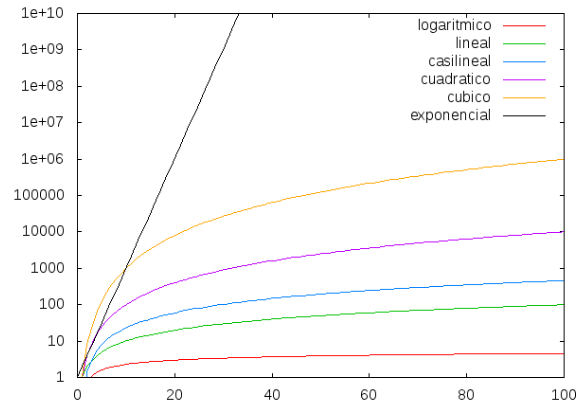
A continuación, se enumeran las propiedades básicas de la notación 0:



1. **Reglas de Cálculo de Complejidad:**

* El tiempo de ejecución de cada sentencia simple, por lo común puede tomarse como . Algunas operaciones matemáticas no deben ser tratadas como operaciones elementales. Por ejemplo, el tiempo necesario para realizar sumas y productos crece con la longitud de los operandos. Sin embargo, en la práctica se consideran elementales siempre que los datos que se usen tengan un tamaño razonable.
* El tiempo de ejecución de una secuencia de proposiciones se determina por la regla de la suma. Es el máximo tiempo de ejecución de una proposición de la sentencia.
* Para las sentencias de bifurcación (IF, CASE) el orden resultante será el de la bifurcación con mayor orden.
* Para los bucles es el orden del cuerpo del bucle sumado tantas veces como se ejecute el bucle. Este tiempo debería incluir: el tiempo propio del cuerpo y el asociado a la evaluación de la condición y su actualización, en su caso. Si se trata de una condición sencilla (sin llamadas a función) el tiempo es , que no es relevante para al cálculo asintótico. Cuando se trata de un bucle donde todas las iteraciones son iguales, entonces el tiempo total será el producto del número de iteraciones por el tiempo que requiere cada una.
* Para los bucles anidados el orden corresponde a la multiplicación del orden del bucle interno por el orden del bucle externo.
* El orden de una llamada a un subprograma no recursivo es el orden del subprograma. (Diaz, N., 2014).
* Algunos de órdenes de complejidad que se presentan con frecuencia son:
* Lineal:
* Cuadrático:
* Polinómico: con que pertenece a los naturales.
* Logarítmico:
* Exponencial:

En la figura 1 se presenta una comparación entre los principales órdenes de complejidad. Es de destacar el excesivo costo que representa una complejidad de orden , frente a la estabilidad de la complejidad logarítmica.



**Imagen 1.** *Aritmética de notación O.*

1. Ejemplo.

En el ejemplo se utiliza un bucle for para recorrer la matriz y encontrar el máximo elemento. La notación O para este algoritmo es O(n), donde n es el tamaño de la matriz. Esto significa que el tiempo de ejecución crece linealmente con respecto al número de elementos en la matriz.

main.cpp

#include <iostream>  
#include <vector>  
  
// Función para encontrar el máximo elemento en una matriz  
**int** encontrarMaximo(**const** std::vector<**int**>& arr) {  
 **int** maxElemento = arr.at(0);  
 **for** (**int** i = 1; i < arr.size(); ++i) {  
 **if** (arr.at(i) > maxElemento) {  
 maxElemento = arr.at(i);  
 }  
 }  
 **return** maxElemento;  
}  
  
**int** main() {  
 std::vector<**int**> numeros = { 10, 5, 8, 15, 3, 20, 12 };  
 **int** maximoElemento = encontrarMaximo(numeros);  
 std::cout << "El máximo elemento es: " << maximoElemento << std::endl;  
 **return** 0;  
}

1. Referencias.

- Diaz, N., (03 de febrero 2014). Introducción a la Complejidad Computacional. Aritmética de Notación O (O grande), <http://artemisa.unicauca.edu.co/~nediaz/EDDI/cap01.htm>.