VAZQUEZ BLANCAS CESAR 2CM7
SAID

ORDEN DE COMPLEJIDAD

ALGORITMOS:

Se llaman *algoritmos* el conjunto de instrucciones sistemáticas y previamente definidas que se utilizan para realizar una determinada tarea. Estas instrucciones están ordenadas y acotadas a manera de pasos a seguir para alcanzar un objetivo.

Todo algoritmo tiene una entrada, conocida como y una salida, y entre medias, están las instrucciones o secuencia de pasos a seguir. Estos pasos deben estar ordenados y, sobre todo, deben ser una serie finita de operaciones que permitan conseguir una determinada solución.

En el mundo de la programación, todo programa o sistema operativo funciona a través de algoritmos, escritos en un lenguaje de programación que el ordenador pueda entender para ejecutar los pasos o instrucciones de una forma automatizada.

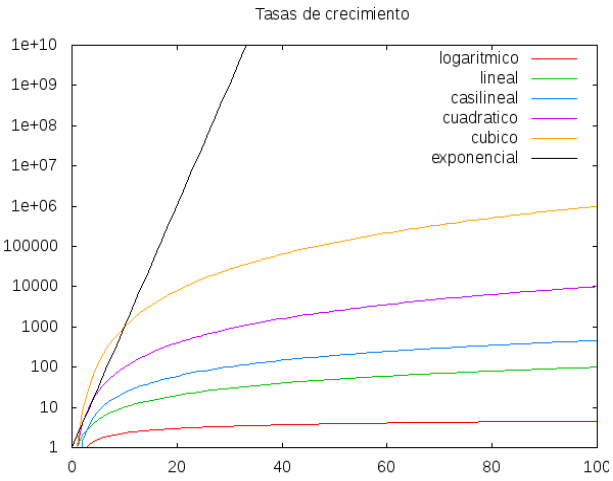
MODELOS MATEMATICOS

Cuando hablamos de complejidad de los algoritmos hablamos principalmente de dos conceptos:

- La complejidad en si que es para un tamaño n tardará un tiempo y para un tiempo mayor cumplirá f(n2) la complejidad nos describe el tipo de curva que cumplirá esa función f. Esto lo representamos como O(f).
- El tiempo que tardará para un tamaño n es llamado tiempo de ejecución. Esto es lo representamos como T(n).

Principales órdenes de complejidad

VAZQUEZ BLANCAS CESAR 2CM7
SAID



Orden	Nombre
O(1)	constante
O(log n)	logarítmica
O(n)	lineal
O(n log n)	casi lineal
O(n^2)	cuadrática
O(n^3)	cúbica
O(a^n)	exponencia

Efectos de duplicaciones:

T(n)	n = 100	n = 200
log(n)	1 h.	1.15 h.
n	1 h.	2 h.
nlog(n)	1 h.	2.30 h.
n^2	1 h.	4 h.
n^3	1 h.	8 h.
2^n	1 h.	$1.27*10^{30}$ h.

T(n)	t = 1h	t = 2h
log(n)	n = 100	n = 10000
n	n = 100	n = 200
nlog(n)	n = 100	n = 178
n^2	n = 100	n = 141
n^3	n = 100	n = 126
2^n	n = 100	n = 101

Recordemos que la complejidad de un algoritmo se puede medir de dos maneras distintas:

– Complejidad temporal: Es la cantidad de operaciones que se realizan en un algoritmo para resolver un problema.

– Complejidad espacial: Es la cantidad de operaciones que se realizan en un algoritmo para resolver un problema, pero en una dimensión dada.

Una notación asintótica es una notación que se usa para representar matemáticamente el comportamiento de una función, pero en las ciencias de la computación se utilizan principalmente para tener una aproximación de la complejidad temporal o espacial de un algoritmo, de esta manera nos podemos hacer una idea de la eficiencia de nuestros programas.

Aproximaciones de tilde. Las aproximaciones de tilde se usan para eliminar términos que pueden complicar las fórmulas.

function	tilde approximation	order of growth
$N^3/6 - N^2/2 + N/3$	$\sim N^3/6$	N^3
$N^2/2 - N/2$	$\sim N^2/2$	N^2
$\lg N + 1$	$\sim \lg N$	$\lg N$
3	~ 3	1

ANALISIS DE ALGORITMOS

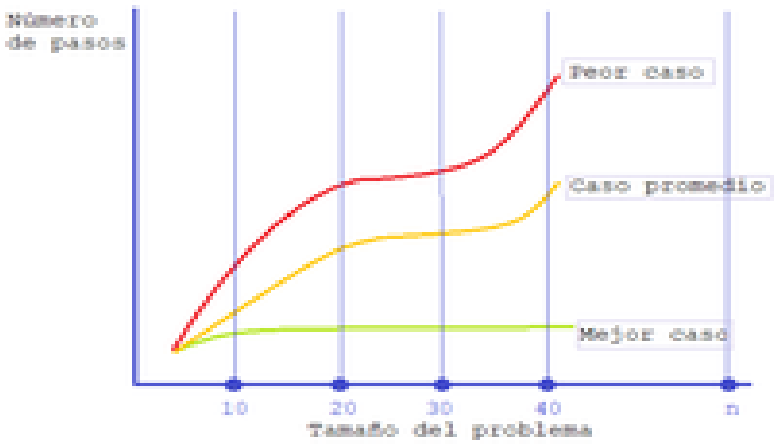
Se refiere al proceso de encontrar la complejidad computacional de un algoritmo que resuelva un problema computacional dado, con el objetivo de proveer estimaciones teóricas de los recursos que necesita. Usualmente, los recursos a los cuales se hace referencia son el tiempo (complejidad temporal) y el almacenamiento (complejidad espacial). Mientras que la complejidad temporal involucra determinar una función que relaciona la longitud o el tamaño de la entrada del algoritmo con el número de pasos que realiza, la complejidad espacial busca la cantidad de ubicaciones de almacenamiento que utiliza.

Distintos algoritmos pueden utilizarse para resolver un mismo problema y a su vez los algoritmos pueden estudiarse de forma independiente del lenguaje de programación a utilizar y de la máquina donde se ejecutará. Esto significa que se necesitan técnicas que permitan comparar la eficiencia de los algoritmos antes de su implementación.

Análisis de los distintos casos
Diferentes entradas de la misma longitud pueden causar que el algoritmo se comporte distinto, por lo que se podría analizar el algoritmo desde tres perspectivas: el mejor caso, el caso promedio y el peor caso.

- **Mejor caso**: es la función definida por el número mínimo de pasos dados en cualquier instancia de tamaño n. Representa la curva más baja en el gráfico (verde) y se denomina cota inferior.
- **Caso promedio:** es la función definida por el número promedio de pasos dados en cualquier instancia de tamaño n.

- **Peor caso**: es la función definida por el número máximo de pasos dados en cualquier instancia de tamaño n. Esto representa la curva que pasa por el punto más alto en el gráfico (rojo) y se denomina cota superior.

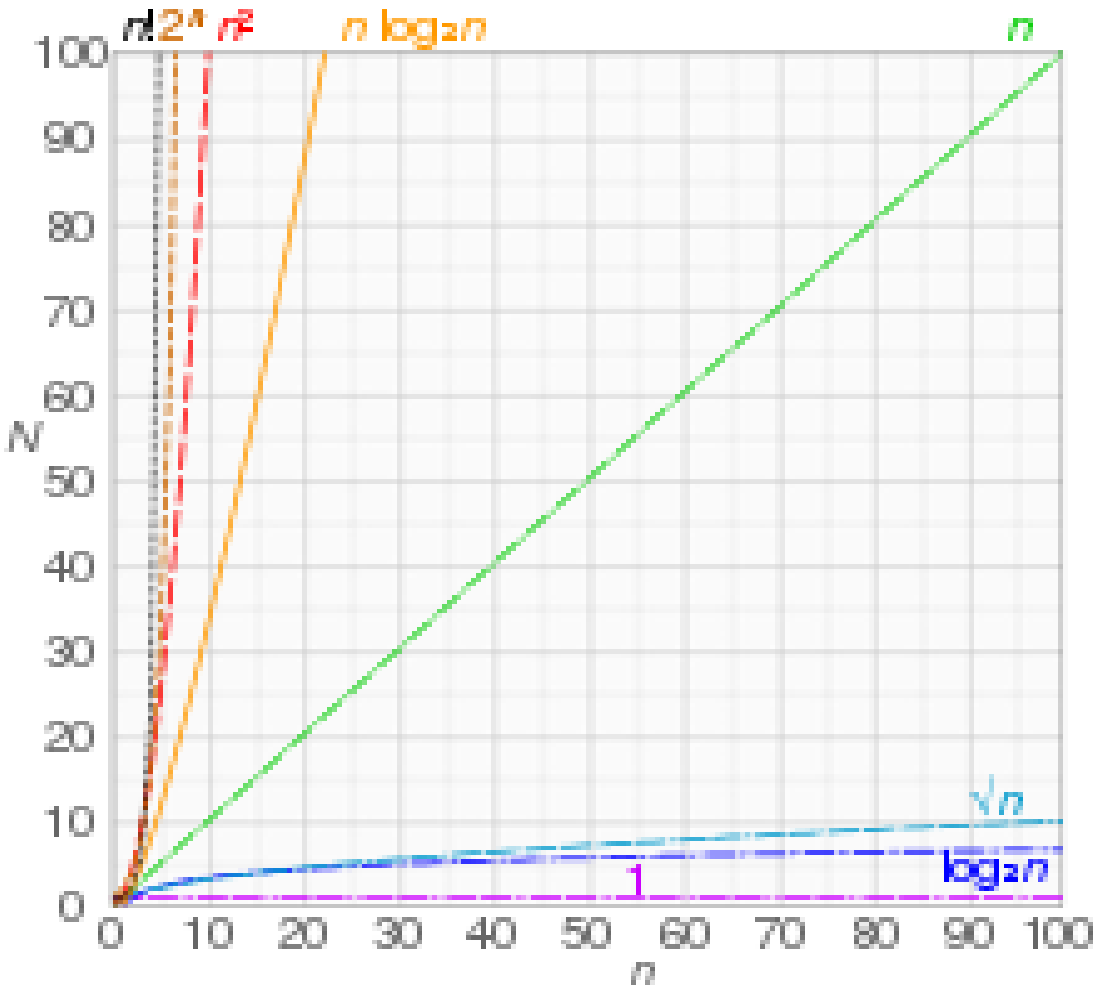


Cuando no se especifica lo contrario, la función que describe el rendimiento de un algoritmo suele este caso, dado que este caso garantiza que el algoritmo no tardará mayor cantidad de tiempo, es decir acota superiormente la cantidad de pasos.

ÓRDENES DE CRECIMIENTO

De manera informal, se puede decir que un algoritmo exhibe una tasa de crecimiento del orden de una función matemática si más allá de un cierto tamaño de entrada n la función f(n) multiplicada por una constante positiva proporciona un límite superior o límite para el tiempo de ejecución de ese algoritmo. En otras palabras, para un tamaño de entrada dado n0 mayor que algún y una constante el tiempo de ejecución de ese algoritmo nunca será mayor que f(n). Este concepto se expresa con frecuencia utilizando la notación O Grande, que brinda una forma conveniente de expresar el peor de los casos para un algoritmo dado.

VAZQUEZ BLANCAS CESAR 2CM7
SAID



:

El orden de crecimiento de un algoritmo es una aproximación del tiempo requerido para ejecutar un programa de computadora a medida que aumenta el tamaño de entrada. El orden de crecimiento ignora el factor constante necesario para las operaciones fijas y, en cambio, se enfoca en las operaciones que aumentan proporcionalmente al tamaño de entrada.

Por ejemplo, un programa con un orden de crecimiento lineal generalmente requiere el doble de tiempo si la entrada se duplica.

Los órdenes de Crecimiento más comunes son:

– Tiempo constante: Cuando un algoritmo tiene un orden de crecimiento constante, significa que siempre toma un número fijo de pasos, sin importar cuánto aumente el tamaño de entrada.

– Tiempo logarítmico: Cuando un algoritmo tiene un orden de crecimiento logarítmico, aumenta proporcionalmente al logaritmo del tamaño de entrada.

– Tiempo lineal: Cuando un algoritmo tiene un orden de crecimiento lineal, su número de pasos aumenta en proporción directa al tamaño de entrada.

– Tiempo cuadrático: Cuando un algoritmo tiene un orden de crecimiento cuadrático, sus pasos aumentan en proporción al tamaño de entrada al cuadrado.

– Tiempo exponencial: Cuando un algoritmo tiene un orden de crecimiento superpolinomial, su número de pasos aumenta más rápido que una función polinomial del tamaño de entrada.