**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Análisis de Algoritmos**

**Ejercicio 3: Graficación de Órdenes de Complejidad**

**Sampayo Hernández Mauro**



**Grupo:** 3CM2

**Profesor:** Edgardo Adrián Franco Martínez

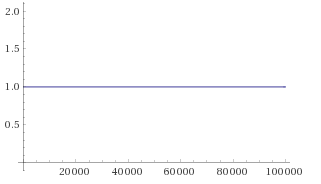
**Fecha de entrega:** 27 de septiembre de 2018

**Ejercicio 3: Graficación de Órdenes de Complejidad**

1. **Dados los órdenes de complejidad, graficar cada uno de estos de manera separada para un rango de 0<n<100,000**

* **O(1) Complejidad Constante.**

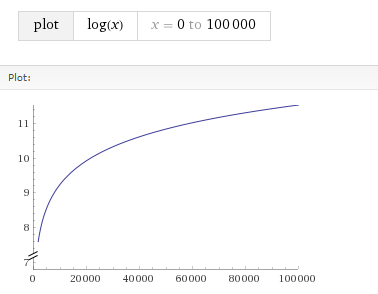
A medida que el tamaño del problema *n* aumente, la complejidad seguirá siendo la misma, es decir, se seguirán realizando la misma cantidad de operaciones.

**Ejemplo:** Una Tabla Hash Idónea.

* **O(log(n)) Complejidad Logarítmica.**

El modelo de crecimiento de la complejidad logarítmica corresponde a la función f(n)=log2(n), es decir, que a medida que el tamaño del problema aumenta, la curva de crecimiento de la cantidad de operaciones crece muy poco. Esto se debe a que, en general, los algoritmos que tienen esta complejidad son aquellos que con una sola operación van descartando grandes porciones de su tamaño de problema.

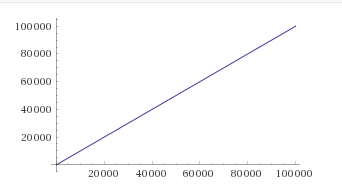
**Ejemplo:** Tree Sort, Búsqueda Binaria.



* **O(n) Complejidad Lineal.**

Modela el comportamiento de algoritmos cuya complejidad crece proporcionalmente al tamaño del problema.

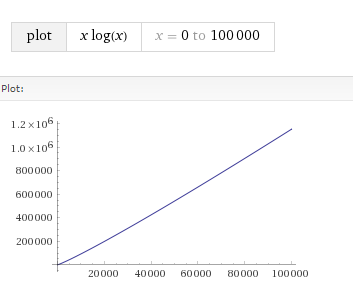
**Ejemplo:** Búsqueda lineal, Radix Sort.



* **O(nlog(n)) Complejidad “n log n”**

Modela el comportamiento de algoritmos que emplean operaciones que dependen directamente del tamaño del problema en combinación con métodos que con una sola operación van descartando grandes porciones de su tamaño de problema. Tienen un crecimiento ligeramente mayor al lineal.

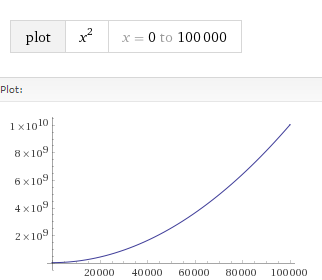
**Ejemplo:** Merge Sort.

****

* **O(n2) Complejidad cuadrática.**

Es la cota de un algoritmo que necesita un desarrollo bidimensional respecto al tamaño del problema *n* para generar una solución. Su curva de crecimiento es notoriamente mayor que las anteriores, por lo que resulta ser ineficiente para tamaños d problema *n* muy grandes.

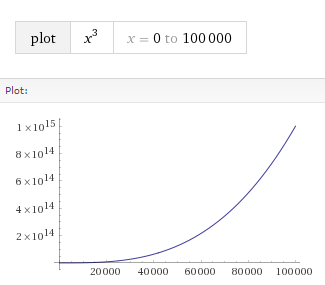
**Ejemplo:** Bubble Sort.



* **O(n3) Complejidad cúbica.**

Describe el comportamiento para algoritmos que requieren un análisis tridimensional respecto a su tamaño de problema, lo que implica aplicar n veces operaciones a conjuntos de operaciones con complejidad n2.

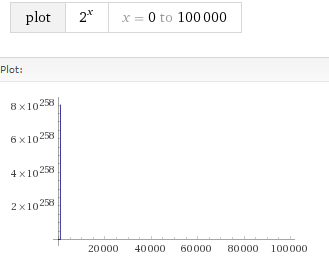
**Ejemplo:** Multiplicación de matrices.



* **O(Cn) Complejidad exponencial.**

Modela el comportamiento de un algoritmo que para un tamaño de problema *n*, necesita *n* capas de C operaciones para encontrar un resultado. Este tipo de algoritmos se consideran sin solución para n muy grandes, es decir, que el tiempo que tardan en llegar a la solución final es excesivamente alto.

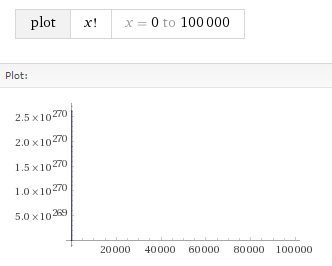
**Ejemplo:** La definición recursiva de las torres de Hanói.



* **O(n!) Complejidad factorial.**

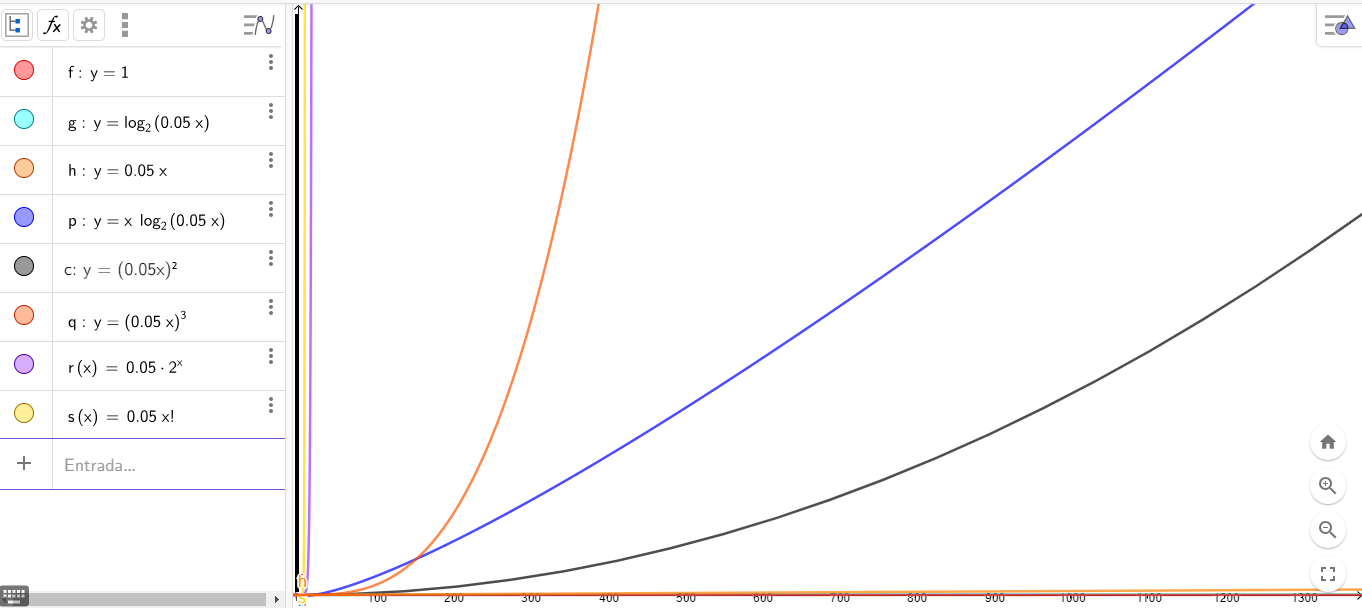
Describe el comportamiento de algoritmos que sirven para encontrar un conjunto de soluciones para un tamaño de problema *n*, analizando caso por caso. Para *n* muy grandes, resulta ser un problema NP.

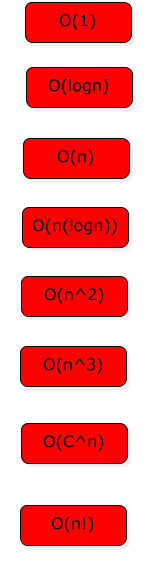
**Ejemplo:** Backtracking.

****

1. **Confronte en pares a todos los órdenes en un rango de 0 < n < 1000 y dé una justificación de cuál elegiría según cada par confrontado.**

Dado a que la confrontación a pares significaría un total de 56 confrontaciones (Lo cual resulta ineficiente). Se ha optado por un análisis total de todas las funciones y así generar una lista de prioridad, siendo el primer elemento el orden que siempre se buscaría, y el último elemento el orden menos deseado.



Podemos concluir que se debe procurar buscar una complejidad constante o de orden nlog(n) debido a que resultan ser las más eficientes. También los ordenes polinómicos pueden resultar ser aceptables, más no deben ser prioritarios debido a que no resultan ser tan eficientes como los dos ordenes de complejidad mencionados anteriormente. Finalmente se deben de evitar los órdenes exponenciales y factoriales, debido a que resultan ser demasiado ineficientes e irresolvibles en tamaños n muy grandes.