COMPLEJIDAD COMPUTACIONAL: T(n) = C * F(n).

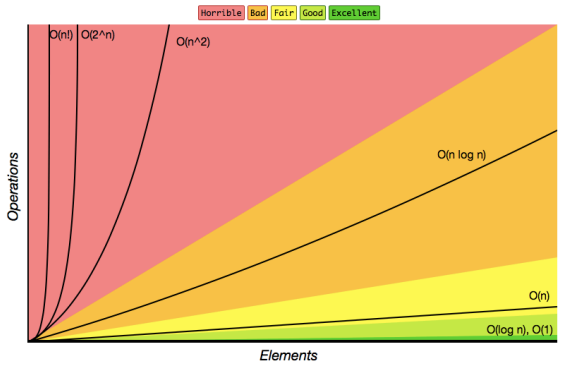
T -> tiempo de respuesta
n -> Tamaño de la muestra
f -> big o
c -> Constante de proporcionalidad. (depende del entorno)

1.2 Propiedades

- Si $g \in O(f)$ y $h \in O(f)$, entonces $g+h \in O(f)$
- Si $f \in O(g)$ y $g \in O(h)$, entonces $f \in O(h)$
- $f+g \in O(\max(f,g))$
- $O(f+g) = O(\max(f,g))$
- Si $f \in O(f')$ y $g \in O(g')$, entonces $f+g \in O(f'+g')$
- Si $f \in O(f')$ y $g \in O(g')$, entonces $f.g \in O(f'.g')$

2.1 Principales órdenes de complejidad

Orden	Nombre
O(1)	constante
O(log n)	logarítmica
O(n)	lineal
O(n log n)	casi lineal
O(n²)	cuadrática
O(n³)	cúbica
O(a^n)	exponencial



2.3 Jerarquía de complejidad

- $O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset O(2^n)$

BIG O Ideal => CTE

Reglas

Regla de la suma. Se toma el mas significativo del conjunto de módulos consecutivos.
Regla del producto. Se multiplican los big O de los módulos anidados.

CTE-> demoran lo mismo sin importar el tamaño de la muestra. Ejemplo encolar y desencolar, apilar y desapilar.
Logarítmica-> búsqueda binaria.
Lineal-> búsqueda secuencial.
NLogN-> QuickSort, cola de prioridad.
Polinómicas -> ordenamiento
Exponencial -> recursividad. No importa la base
Factorial -> recursividad ejemplo factorial