

Eficiencia

De los algoritmos

Eficacia vs Eficiencia

Eficacia es la capacidad de realizar un efecto deseado, esperado o anhelado.

Resolver el problema

eficiencia es la capacidad de lograr ese efecto en cuestión con el mínimo de recursos posibles o en el menor tiempo posible.

Resolver el problema de la mejor manera

Los recursos informáticos que a necesitar

Método para evaluar:

- Independiente del hardware
- Independiente del lenguaje informático
- Independiente a factores externos del algoritmo

La historia del niño Gauss

$$S = 1 + 2 + 3 + 4 + 5 + 6 + 7 \\ \dots + 100$$

Lo que Gauss hizo:

$100 / 2 = 50 \rightarrow$ mitad del conjunto

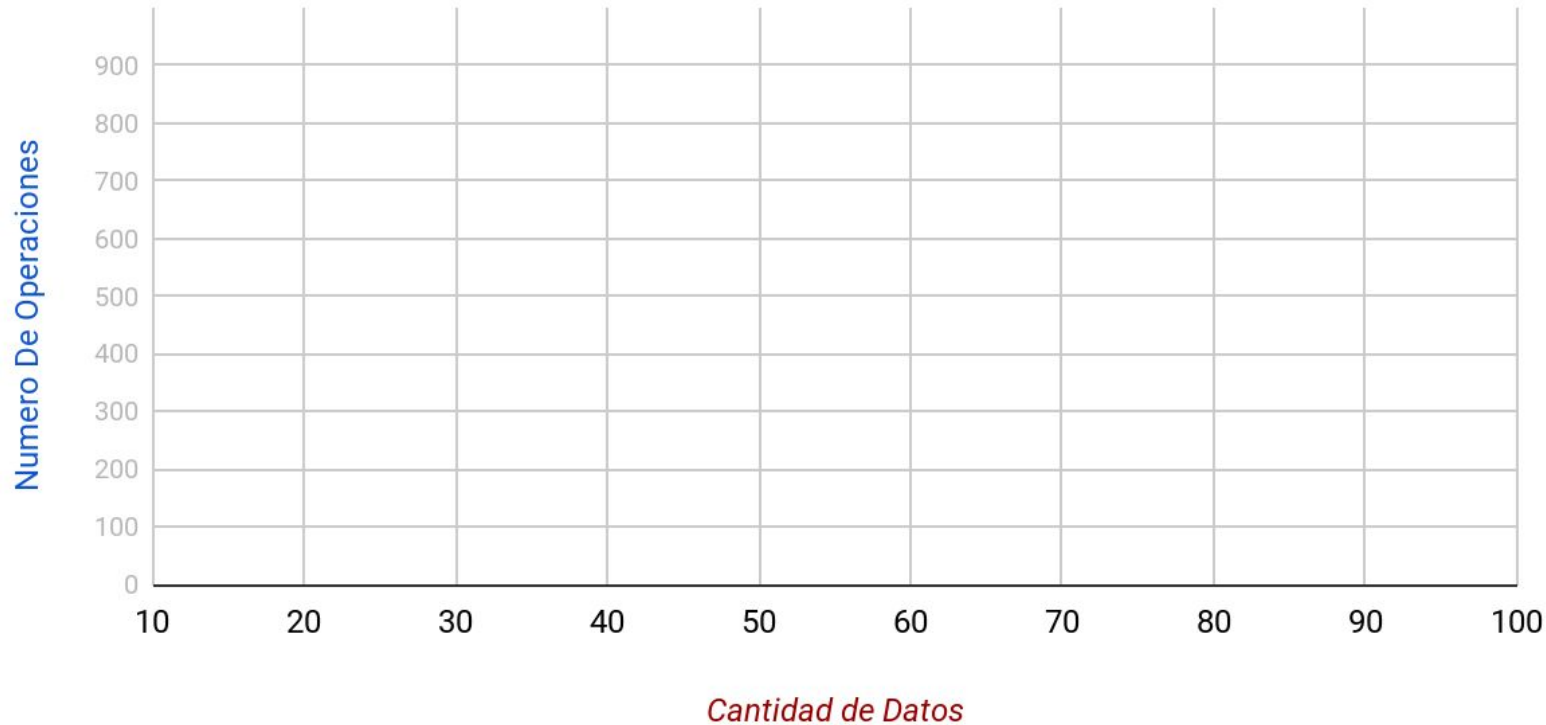
$$\begin{aligned} 1 + 100 &= 101 \\ 2 + 99 &= 101 \\ 3 + 98 &= 101 \\ 4 + 97 &= 101 \\ 5 + 96 &= 101 \\ 6 + 95 &= 101 \\ 7 + 94 &= 101 \\ &\dots \\ 50 + 51 &= 101 \end{aligned}$$

$$S = \frac{n+1}{2} \cdot n$$

$$S = (n/2) * (n+1)$$

No es confiable **usar el tiempo reloj como medida del tiempo de ejecución de los algoritmos**. Esto se debe a que en una misma computadora para un mismo algoritmo, el tiempo reloj **puede variar bastante dependiendo** de diversos factores, como lo son **el sistema operativo** o la **ejecución** de ciertas **interrupciones o procesos** que se salen del control del usuario. Estas variaciones pueden ser bastante más significativas si se compara el tiempo de ejecución de programas en máquinas con **capacidad distinta o con arquitecturas diferentes**. Por ello la trascendencia en el uso de la notación matemática en todos los análisis de algoritmos.

Tasa de crecimiento --> orden de complejidad



Medición del algoritmo

CUÁNTO TIEMPO TARDA? CUÁNTOS RECURSOS CONSUME?

La **notación asintótica** es de suma importancia en ciencias de la computación para **determinar el tiempo de ejecución de los algoritmos** y/o hacer comparaciones entre ellos. Sirve de **parámetro de referencia estándar**, ya que no hay un modelo o máquina universal contra el que se puedan evaluar todos los algoritmos en su tiempo de ejecución o corrimiento.

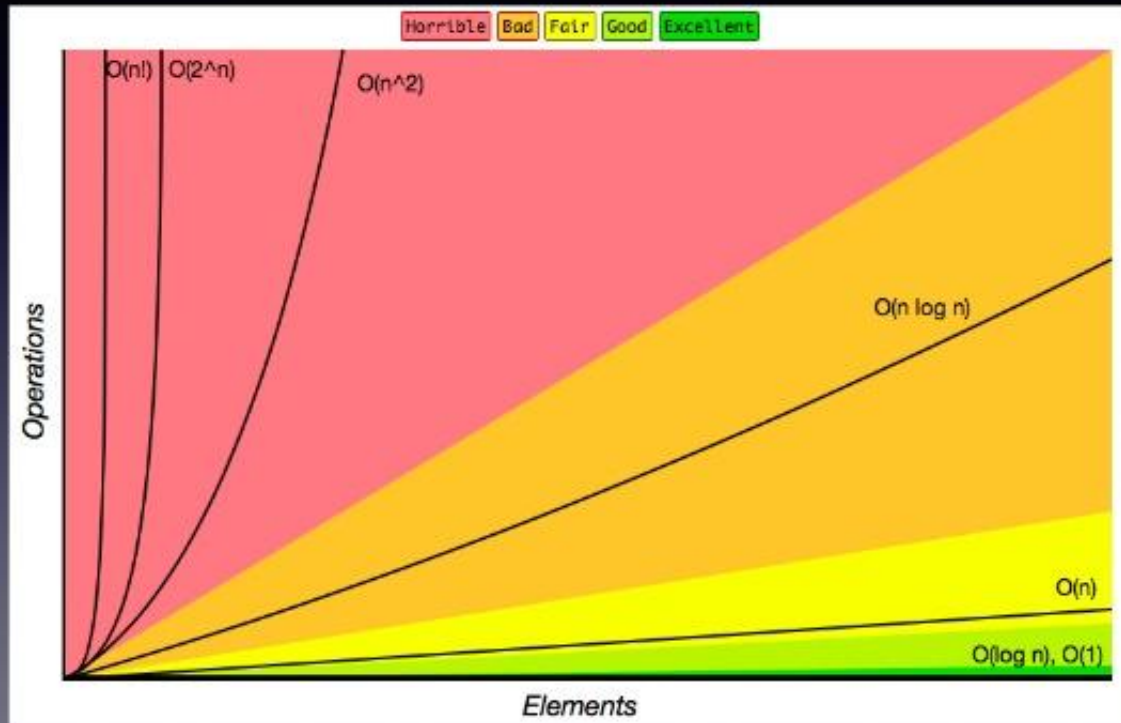
Tipos de complejidad

O Grande (Big O) \Rightarrow Omicron \longrightarrow El peor de los casos

Ω Grande \Rightarrow Omega \longrightarrow El mejor de los casos

Θ Grande \Rightarrow Theta \longrightarrow El caso promedio

Comparison Big O



28

$O(1)$ -> Constante

$O(\log n)$ -> Logaritmica

$O(n \log(n))$

$O(n)$ -> Lineal

$O(n^2)$ -> Cuadratica

$O(n^a)$ -> Polinomial

$O(n^2)$ -> Exponencial

$O(n!)$ -> Combinada

Distintos tipos de complejidad

```
listado = [23, 5, 78, 9, 45, 33, 98, 34, 45, 67]
```

```
dato_buscar = int(input())
```

```
def buscador(numeros, encontrar):  
    for i in numeros:  
        if i == encontrar:  
            Return True  
    return False
```



Ω



Θ

```
if buscador(listado, dato_buscar):  
    print("dato encontrado")  
else:  
    print("no se encontró el dato")
```



O

Busqueda lineal

```

def suma_lineal(cant):
    suma = 0
    for numero in range(1, cant+1):
        suma += numero
    return suma

# 1 + n * 2 ==> 2n
# O(n) lineal

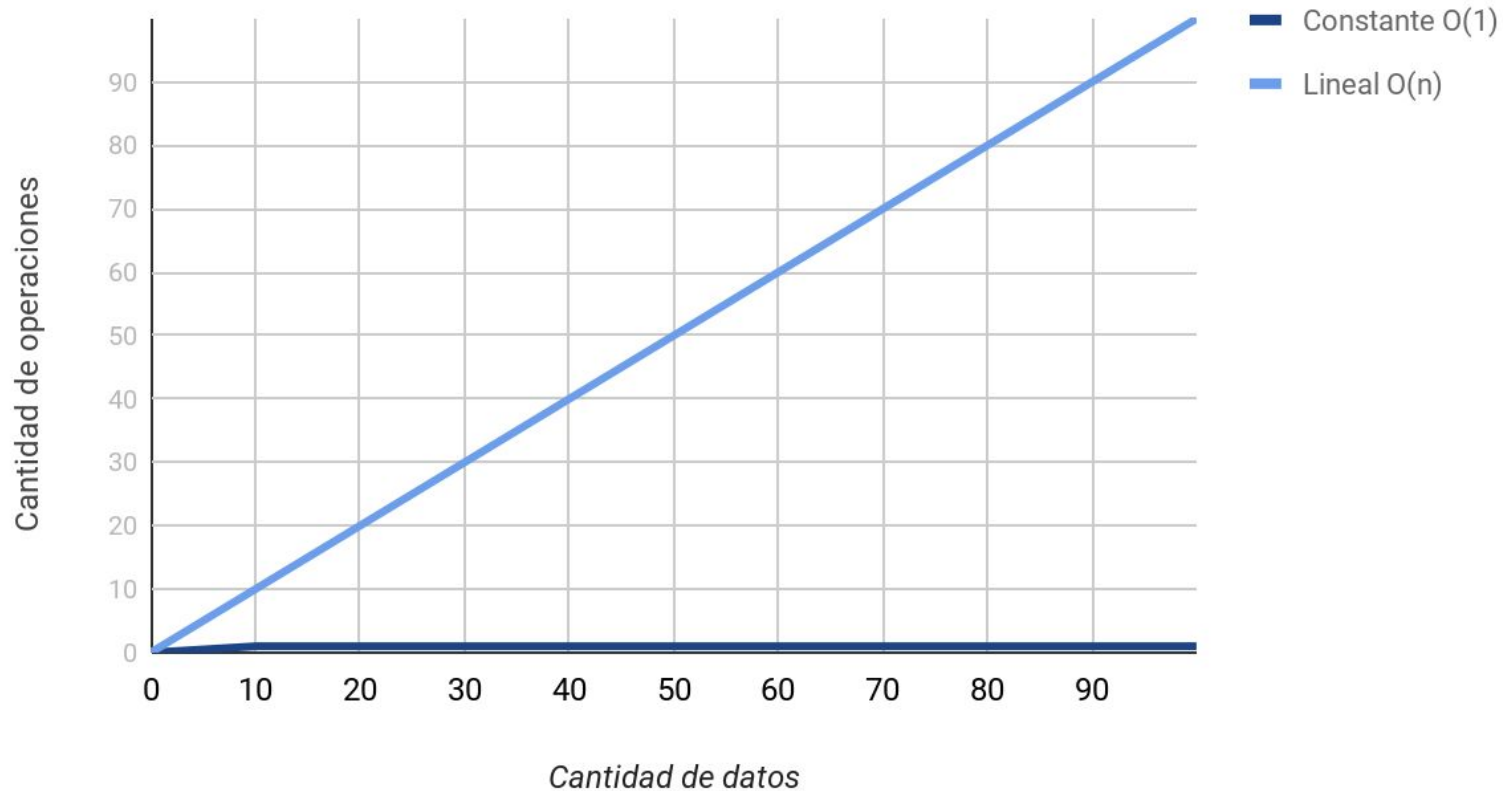
def suma_continua(cant):
    return (cant / 2 ) * (cant +1)

# 1 + 1 + 1 = 3 ==> O(1) constante

```

Analisis: número de operaciones que se llevan a cabo

Sumas









NOTACION ASINTÓTICA / BIG O / O GRANDE

- Es una medida estimada
- Se ignoran las constantes
- Se toma sólo el término más complejo

Complejidad Cuadrática $O(n^2)$

Escribir una función para comprobar que dada la suma de 2 números (pueden ser el mismo) de una lista dan el valor buscado

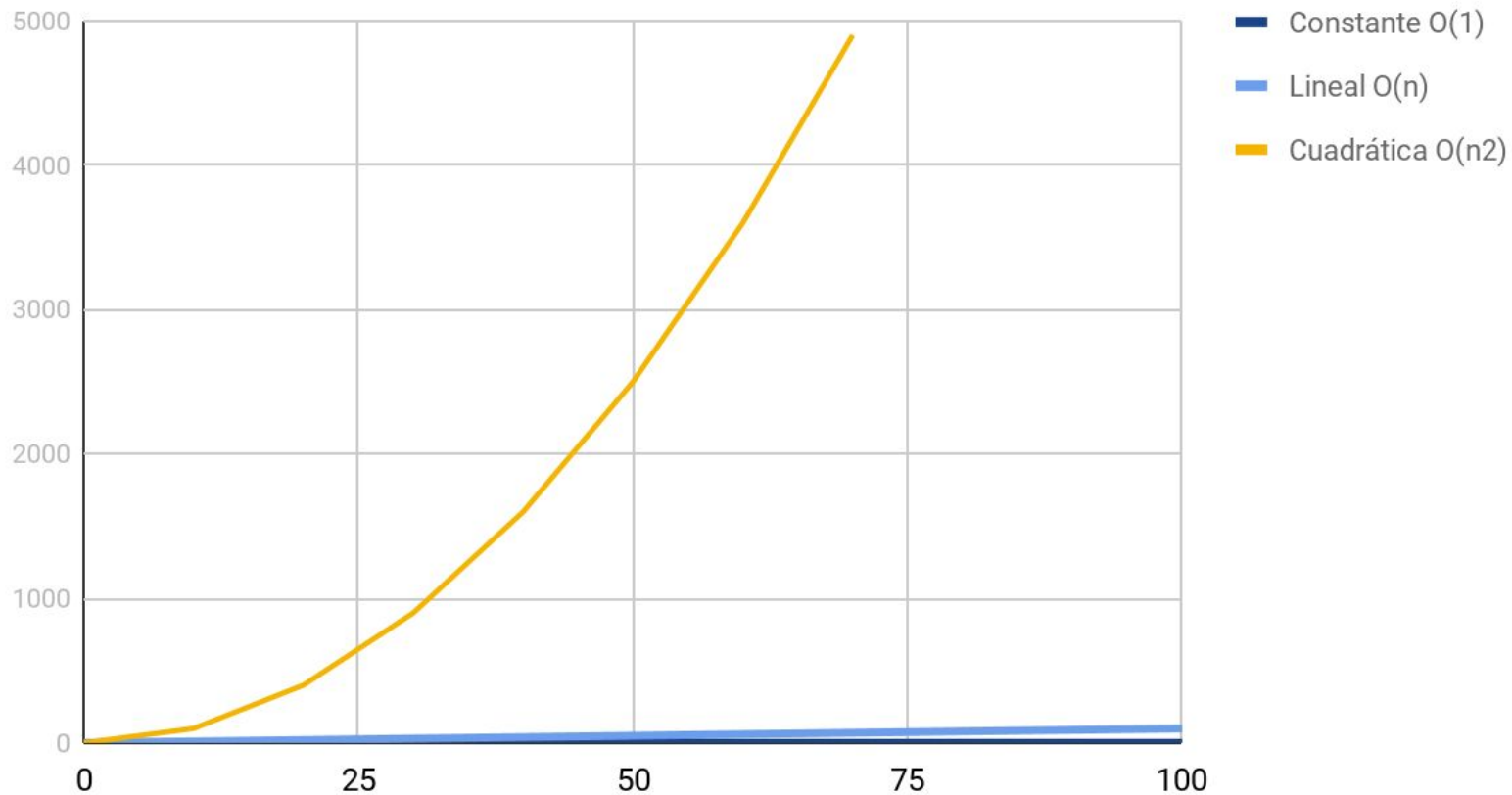
```
def suma_dos(lista, valor):  
    pares = []  CONSTANTE  
    for i in lista:  N  
        for j in lista:  N   $N * N = N^2$   
            if i + j == valor:  
                pares.append([i, j])   CONSTANTE  
    return pares
```

```
cantidad = [12, 23, 41, 36, 28, 54, 35, 46, 52, 19, 27]
```

```
numero = 74
```

```
print(suma_dos(cantidad, numero))
```

Cuadrática



Análisis