

# Notas

## Exploring Data Structures 1.

@CuriesCommunity ❤

Atte. Elisa

# Complejidad? Algoritmo?

- \* Tiempo - HIS
  - \* Dificultad -
  - \* - H
- \* Serie Pasos
  - \* Resolver problema
  - \* Estruct.

"Todo cuesta..."

- \* Eficaz
- \* Específico

\$  
Hardware - Software  
guardar files

↓ costo  
bytes / bit

1 byte = 8 bits

Tiempo  
en  
memoria

M/G/T Storage

Software

comp.  
en  
tiempo

Algoritmo?

↓  
complejidad

↓ costo  
tiempo

\* Serie Pasos

\* Resuelve  
→ problema

\* Estruct.

\* Eficaz

\* Específico

Complejidad

de 1 algoritmo

qué? al costo en tiempo

Para qué?  
util?

eficientes  
eficaces  
muy costoso

tam  
in.

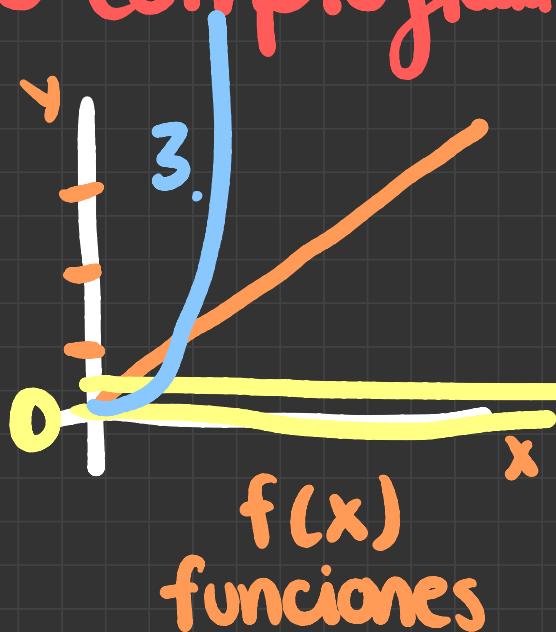
para describir cómo crece  
se comporta el alg. en tiempo  
mem

# Notaciones de complejidad 'acotar'

1. Mejor caso

2. Prom. caso

3. Peor caso



1. Big-Omega  $f(x)$

2. ¿ $\Omega(f(x))$

3. Big-O peor caso  
 $O(n)$

Big-O tratamos evitar ↪ complejidades tipos



# $O(1)$ constante

Sumas  $2+3=5$  imprimir  
asignaciones  $(a=b+c/d)$  fórmula

# $O(\log n)$ logarítmica

búsqueda binaria  
exponenciación bin.

# $O(n)$ lineal

for / while / iteración de arr

# $O(n \log n)$ linearítmica

merge sort  
y paradigma  
divide y vencerás  
divide & conquer

# $O(n^2)$ cuadrática

for ( N ) {  
 for ( N ) {  
 //  
 }  
}

bubble sort  
insertion sort

for ( N ) {  
 for ( N ) {  
 while ( )  
 //  
 }  
}

# $O(n^3)$ cúbica

# $O(2^n)$ exponencial

fuerza bruta

# $O(n!)$ factorial

permutaciones  
combinaciones  $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

# Listas

- EDD
- almacenar d.
- consecutivo

## Qué necesitamos?

L apunadores

L complejidad

variable

int a = 10;

↑ tipo de dato

clase

Persona

Poo



→ verbos

L métodos

L atributos

↓ característica

- nombre

- altura

- edad

- CURP

Pedro

pt.



¿? nodo

Super

persona camina

L. Super → <sup>1</sup> agregar

L Leche

L Huevo

L Tortillas

L Pan



recorrer

buscar

P P  
1 3



eliminar

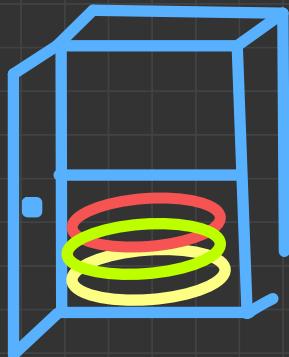
(valor, idx)

L Leche  
L Huevo  
L Tortillas  
L Pan

L Leche  
L Huevo  
L Cafe  
L Pan

consultar\_tam  
eliminar 1 elem  
buscar 1 elem  
agregar en pos x

# Pilas cocina



# Orden comportamiento

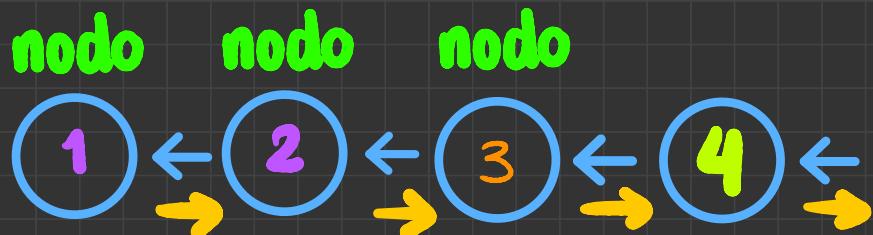
último entrar clífo?

1º salir

last in  
first out

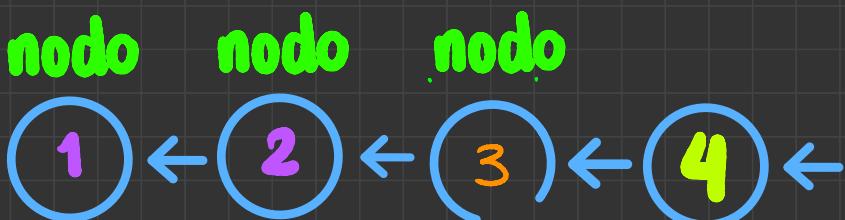
1. root → al tope

arriba  
↳ tope  
top



Sacar

sacar 1  
plato



agregar

colocar platoActual

```
struct nodoMisAmigos  
{  
    string nombre;  
    nodoMisAmigo *sigAmigo;  
    nodoMisAmigo *antAmigo;
```

