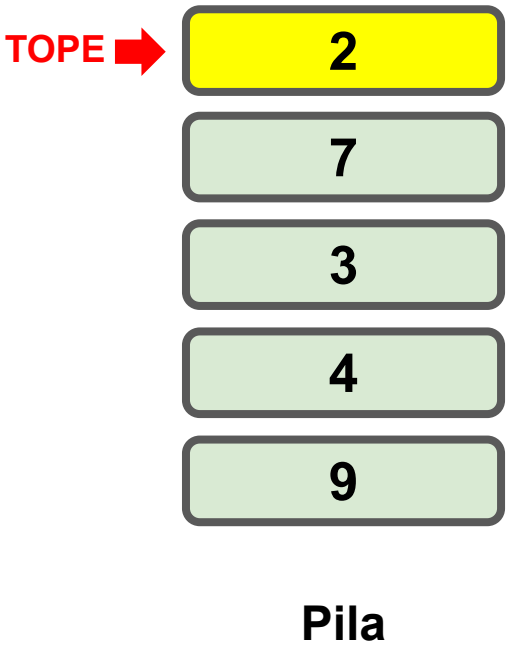
# Pilas

## 75.41 - Algoritmos y Programación II

1° Cuatrimestre 2022

# ¿Qué es una pila?

- Estructura de datos
- Agrupa elementos
- **LIFO** → “Last In, First Out”



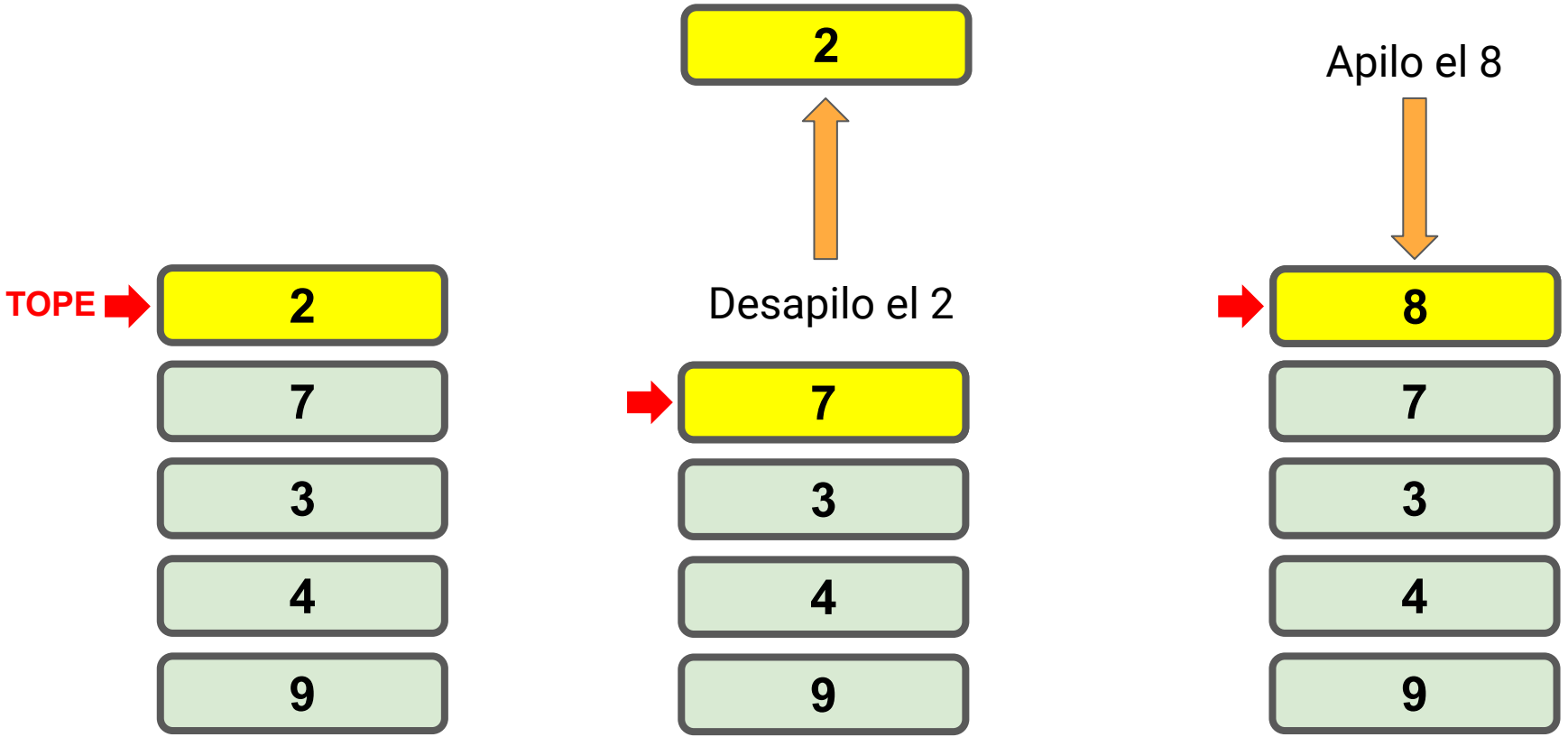
# Operaciones


- Crear (create)
- Apilar (*push*)
- Tope (top)

# Operaciones


- Destruir (destroy)
- Desapilar (*pop*)
- Vacía (isEmpty)

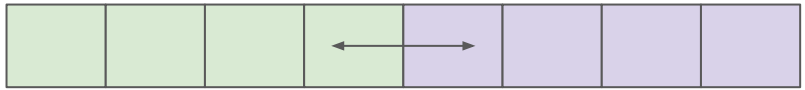
# Ejemplos de operaciones sobre la Pila



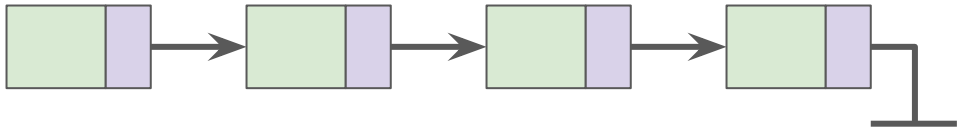
# Implementaciones

- Vector estático

- Vector dinámico



- Pila como lista de nodos



# Primera implementación: ¡Vector estático!


Ehh.

# Primer implementación: Vector estático

PILA:

TOPE: 0 → cantidad de elementos actualmente almacenados

CAPACIDAD: 5 → cantidad de elementos que puedo almacenar

¿Puedo apilar? **SI**. TOPE != CAPACIDAD
¿Está vacía? **SI.** TOPE == 0
¿Puedo desapilar? **NO**. TOPE == 0

PILA:


Apilo el elemento 55

TOPE: 1

CAPACIDAD: 5

¿Puedo apilar? **SI**. TOPE != CAPACIDAD
¿Está vacía? **NO.** TOPE != 0
¿Puedo desapilar? **SI**. TOPE != 0

PILA:


Apilo el elemento 6

TOPE: 2

CAPACIDAD: 5

¿Puedo apilar? **SI**. TOPE != CAPACIDAD
¿Está vacía? **NO.** TOPE != 0
¿Puedo desapilar? **SI**. TOPE != 0

PILA:


Apilo más elementos

TOPE: 5

CAPACIDAD: 5

¿Puedo apilar? **NO**. TOPE == CAPACIDAD
¿Está vacía? **NO.** TOPE != 0
¿Puedo desapilar? **SI**. TOPE != 0

PILA:


Desapilo

TOPE: 4

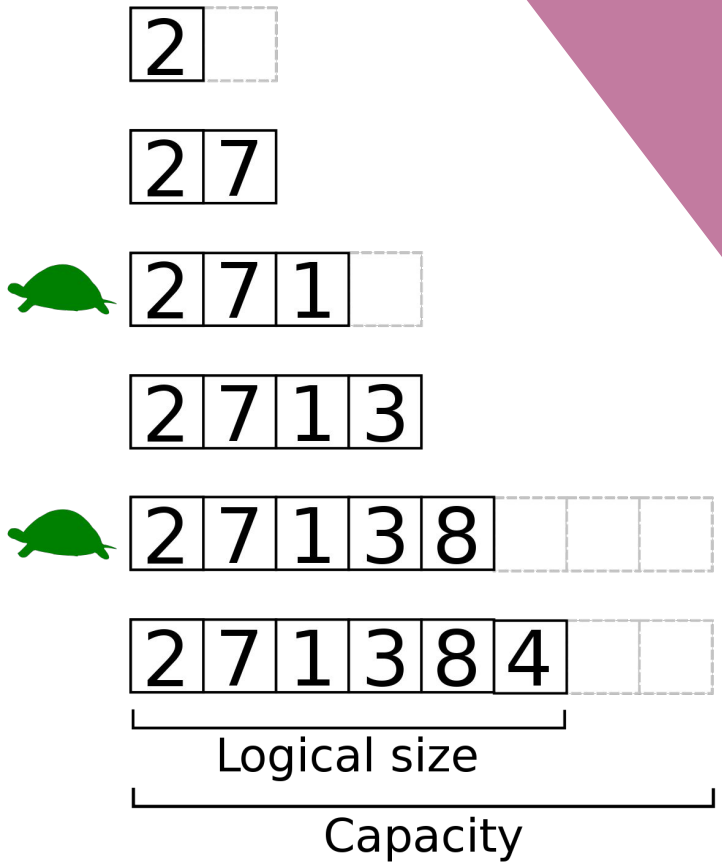
CAPACIDAD: 5

¿Puedo apilar? **SI**. TOPE != CAPACIDAD
¿Está vacía? **NO.** TOPE != 0
¿Puedo desapilar? **SI**. TOPE != 0

```c
typedef struct pila {
    int tope; // cantidad de elementos almacenados
    void* elementos[CAPACIDAD_PILA]; // vector en donde
                                     // se almacenarán los
                                     // elementos
} pila_t;
```

Problema…

¿Qué pasa si quiero almacenar más elementos?




# Segunda implementación: ¡Vector dinámico!

## Pila llena

PILA:


TOPE: 5

TAMANIO: 5

¿Puedo apilar? **¡SI! Puedo pedir más memoria**
¿Está vacía? **NO.** TOPE != 0
¿Puedo desapilar? **SI**. TOPE != 0

¿Cuanto agrandamos el vector?

# Redimensión

- Depende de la implementación. Por ejemplo, la pila crece:
  - Cuando se supera el 75% de la capacidad
  - Cuando se llena
  - Etc
- Cuando desapilo, también redimensiono:
  - Cuando se llega al 50% de la capacidad
  - Cuando desapilo
  - Cuando se llega al 25% de la capacidad
  - Etc

# Repaso de REALLOC

```
void* realloc(
        void* ptr,
        size_t tamanio_nuevo
        );
```

Modifica el tamaño del bloque de memoria apuntado por *ptr* en *tamanio_nuevo* bytes

# A tener en cuenta...

- Sirve siempre y cuando haya memoria **contigua** disponible

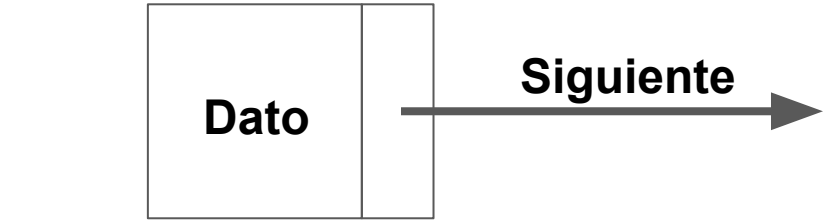
- Sino… ¡no voy a poder redimensionar!

  ¿Y ahora?

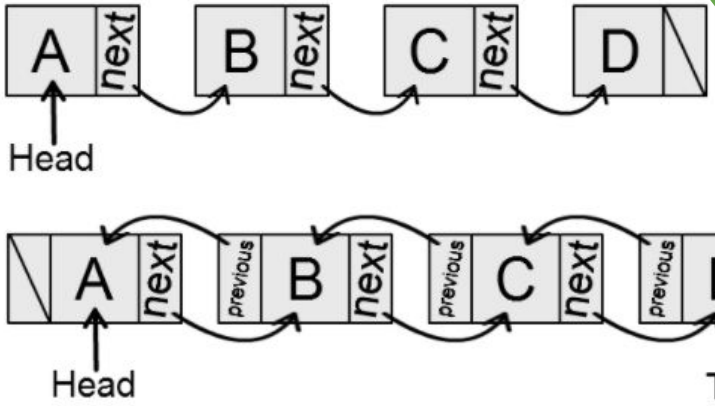
# TDA Nodo enlazado

**¿Que idea abstrae?**

Se trata de abstraer la idea de un contenedor que no solo va a contener a un tipo de dato determinado, sino que también podría apuntar al próximo elemento.




# Tercera implementación: ¡Nodos Enlazados!



# Solución: **lista de nodos**

- Los elementos son **nodos**
- Cada uno tiene una referencia al nodo anterior

¿Cuándo reservo / libero memoria?

- Reservo memoria para cada nodo cuando quiero apilar
- Libero memoria para cada nodo cuando quiero desapilar

Ventaja:

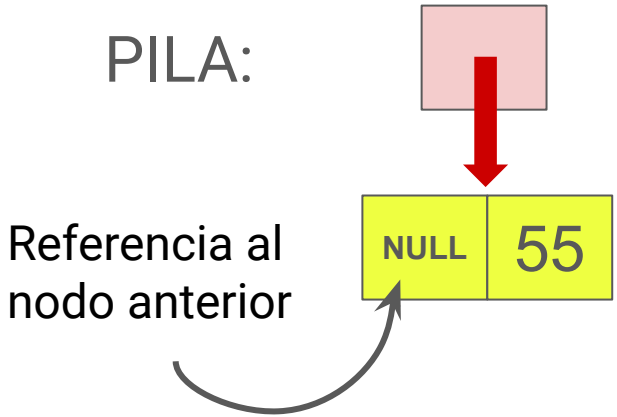
- Memoria **no debe ser contigua**

# Lista de nodos

PILA: NULL

Referencia a nodo_tope es NULL

PILA:

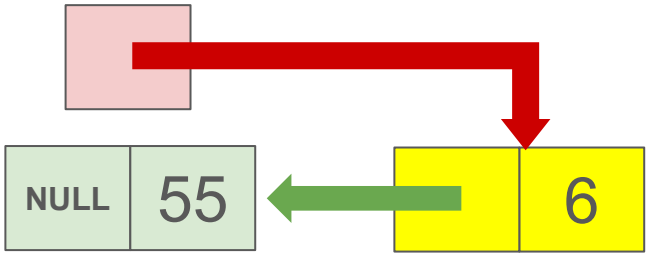



Apilo un elemento

# Lista de nodos

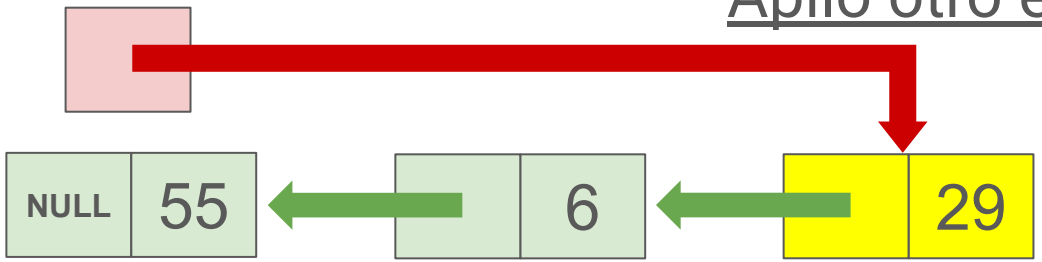
PILA:

Apilo un elemento



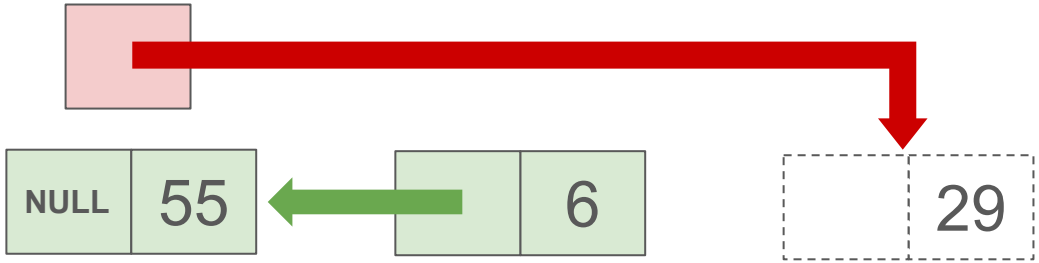

PILA:

Apilo otro elemento



# ¡Cuidado al desapilar!

PILA:

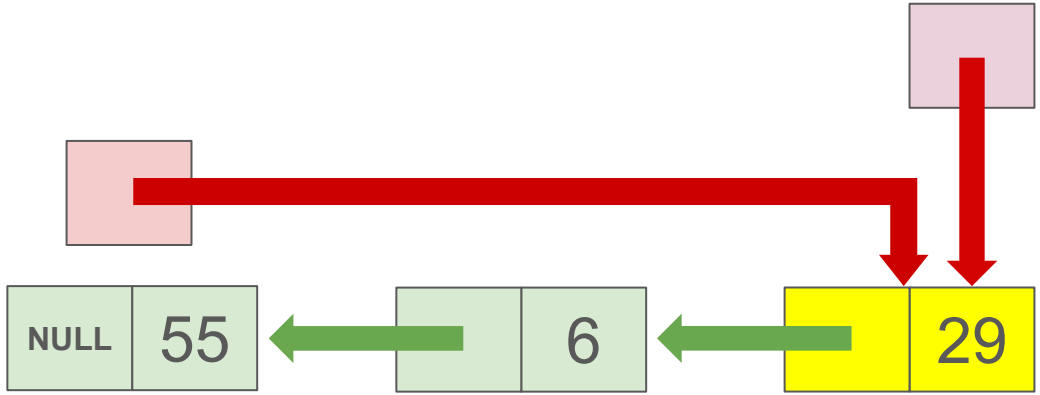



¡Perdí la referencia a los otros nodos!

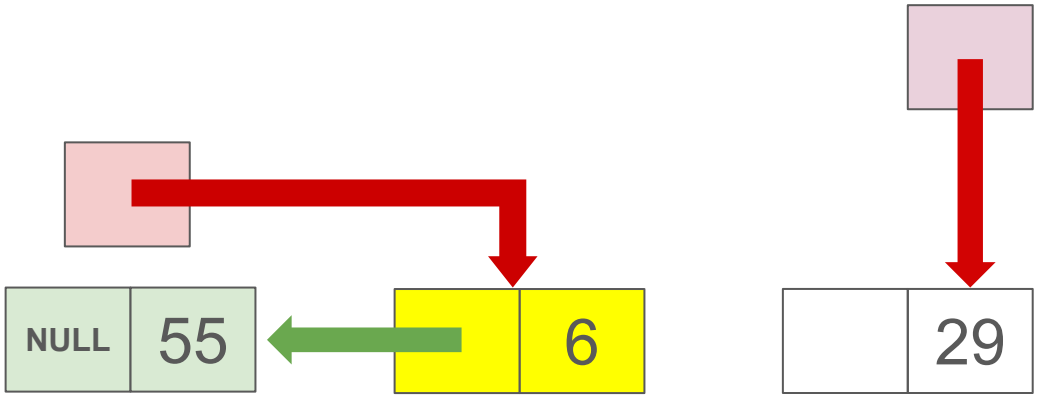
# ¡Cuidado al desapilar!

Nodo a eliminar

PILA:

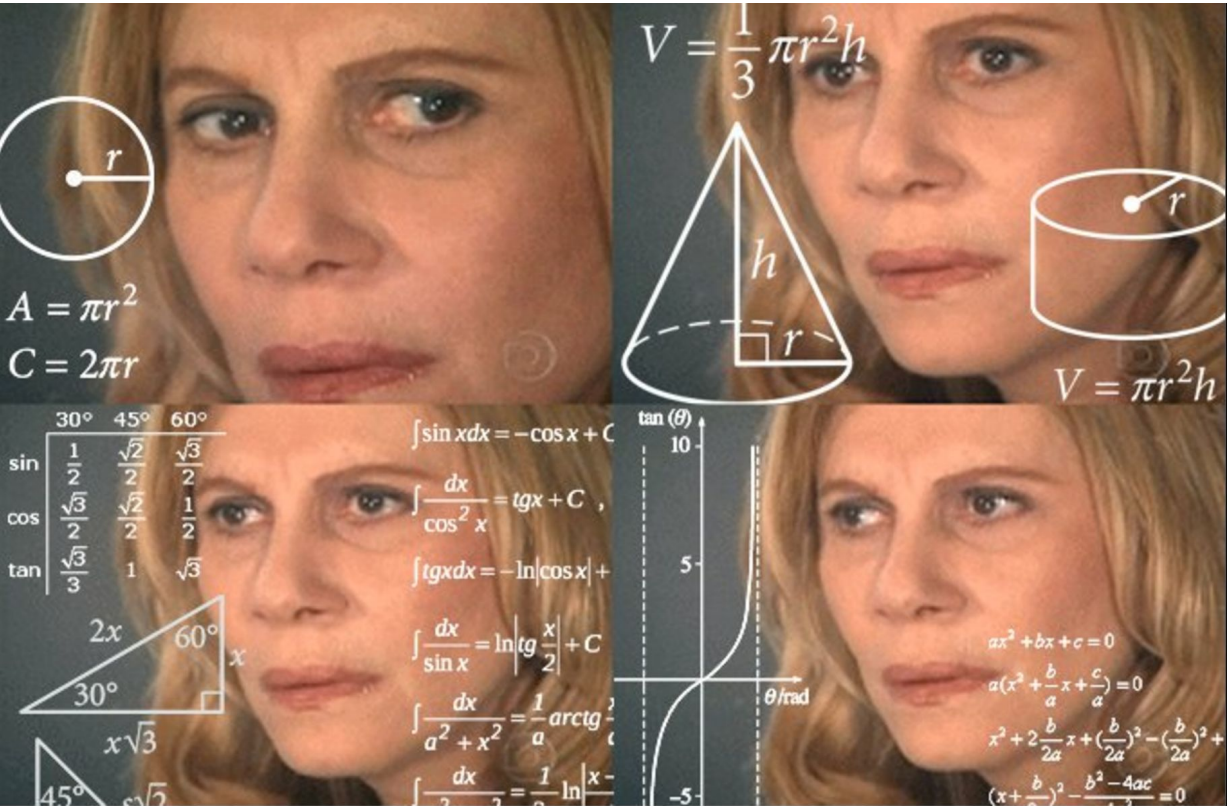



PILA:



# ¡Momento!

¿Y la complejidad?




## Vector Estático

- **¿Crear?**

  **O(1)**

- **¿Destruir?**

  **O(1) / O(n)**

- **¿Push?**

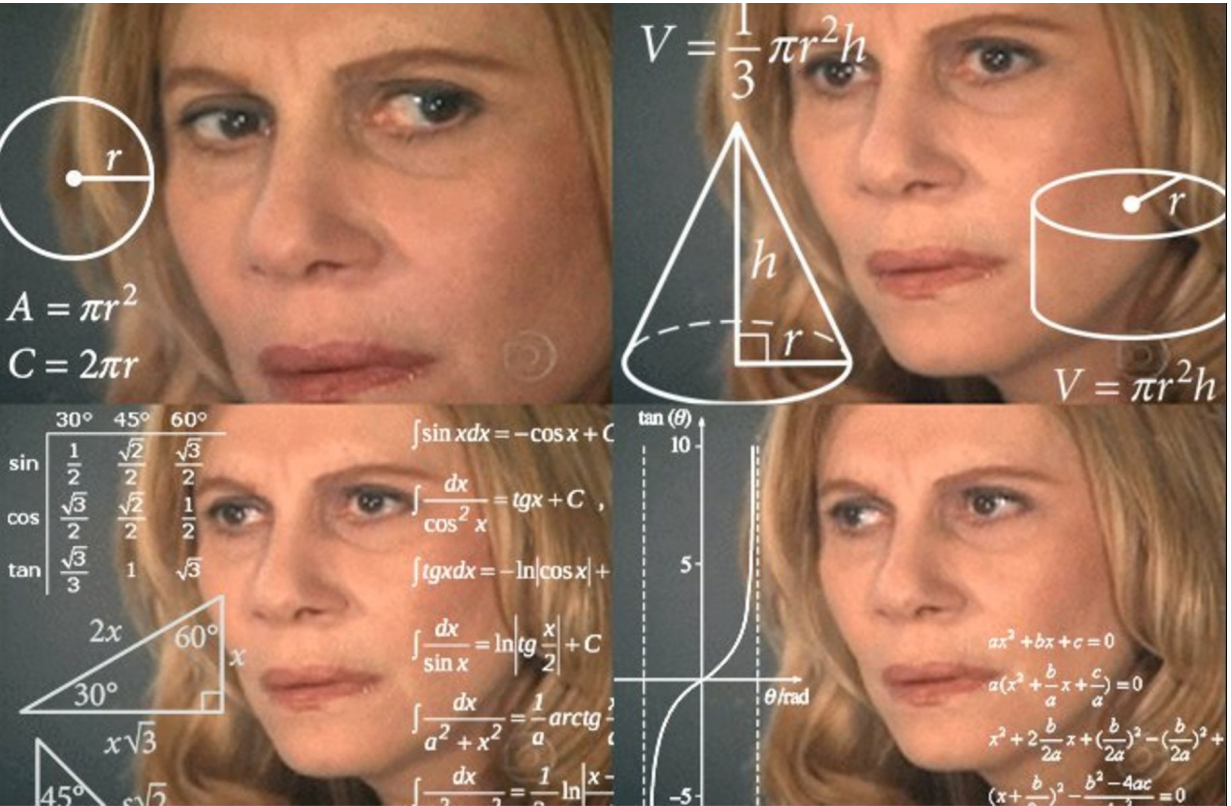
  **O(1)**

- **¿Pop?**

  **O(1)**

# ¡Momento!

¿Y la complejidad?




## Vector Dinámico

- **¿Crear?**

  **O(1)**

- **¿Destruir?**

  **O(1)/O(n)**

- **¿Push?**

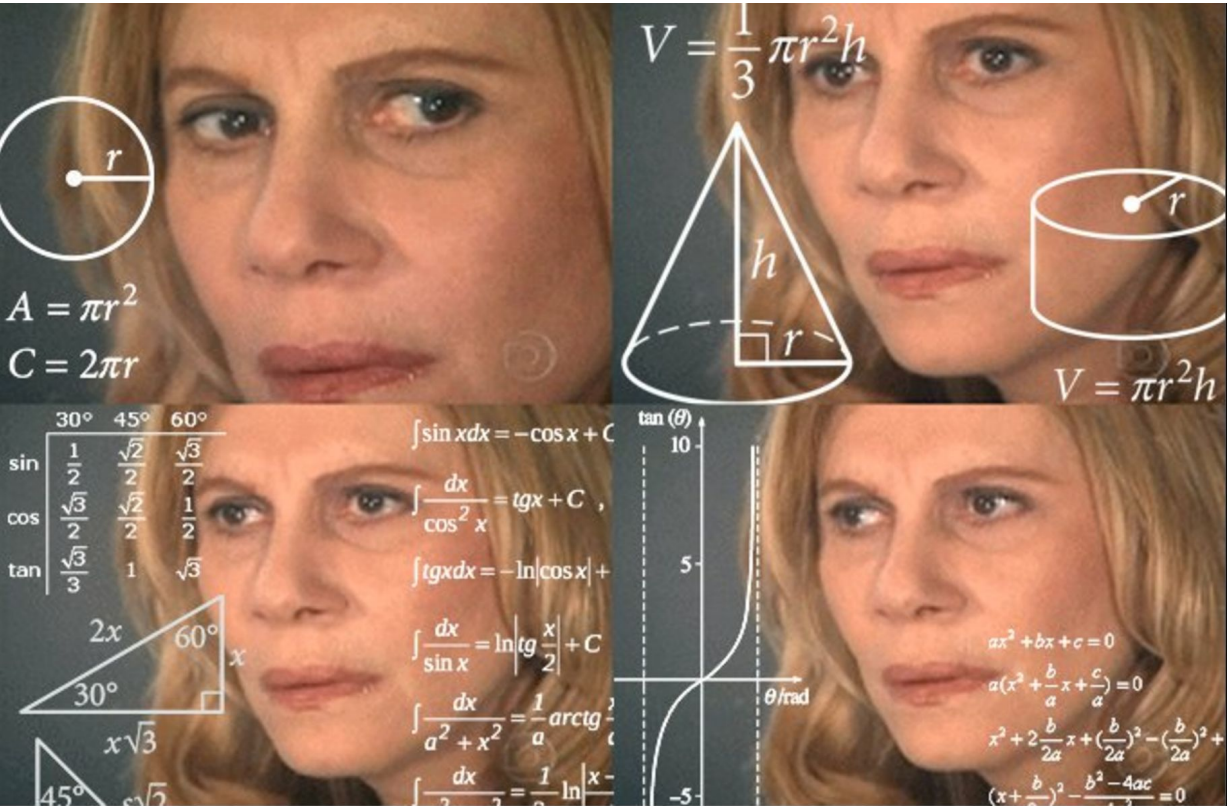
  **O(n)**

- **¿Pop?**

  **O(n)**

# ¡Momento!

¿Y la complejidad?




## Nodos Enlazados

- **¿Crear?**

  **O(1)**

- **¿Destruir?**

  **O(n)**

- **¿Push?**

  **O(1)**

- **¿Pop?**

  **O(1)**


# ¿PREGUNTAS?