



# Análisis de Algoritmos y Estructura de Datos

**Semana 7**  
**Aplicaciones de pilas y colas**

Prof. Violeta Chang C

Semestre 2 – 2023



# Aplicaciones de pilas y colas

- **Contenidos:**

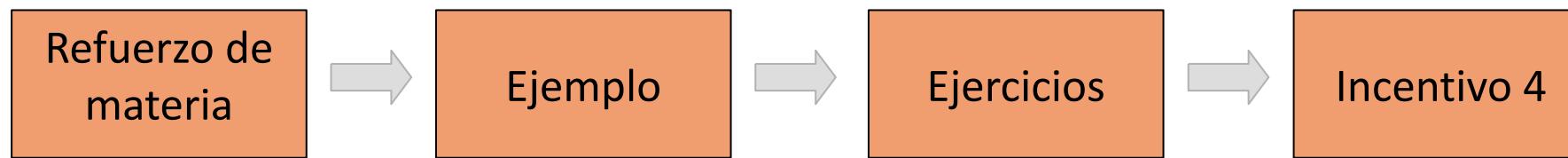
- Operaciones y aplicaciones con pilas
- Operaciones y aplicaciones con colas

- **Objetivos:**

- Conocer aplicaciones clásicas de pilas
- Comprender funcionamiento de operaciones con pilas y aplicar dichas operaciones para resolver problemas específicos
- Conocer aplicaciones clásicas de colas
- Comprender funcionamiento de operaciones con colas y aplicar dichas operaciones para resolver problemas específicos



# Ruta de la sesión





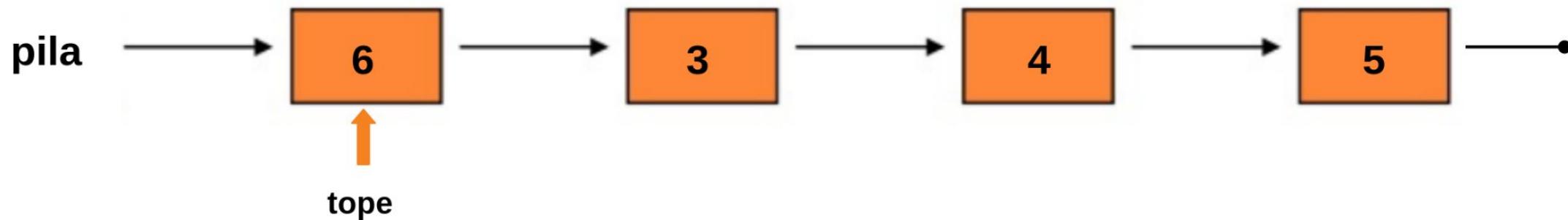
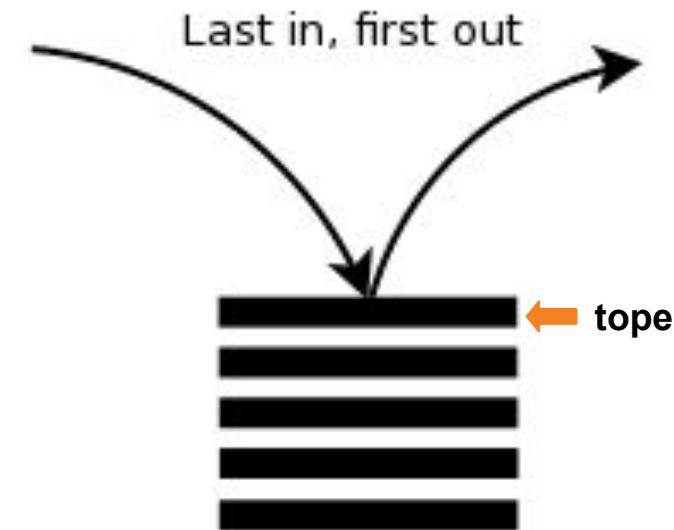
# Refuerzo de materia



# Especificación de TDA pila

- **Estructura de datos:**

- Una pila es una secuencia de elementos que cumple la propiedad **LIFO** (last in, first out), es decir, el último elemento que se introduce en la pila, será el primer elemento en salir de ella





# Especificación de TDA pila

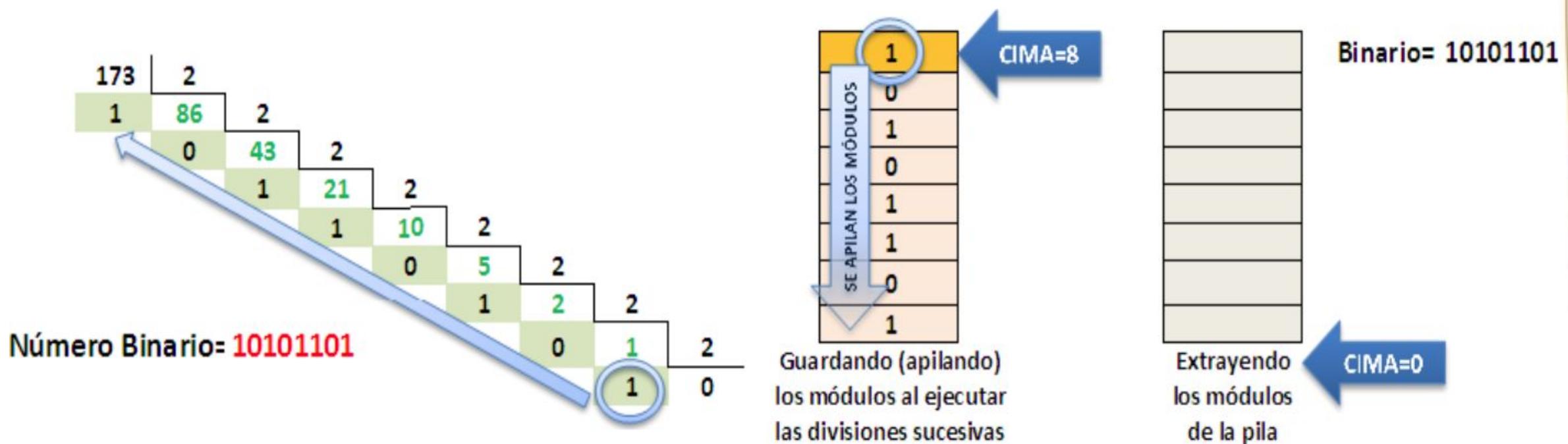
- **Operaciones:**

- **apilar(pila,dato):** agrega un elemento al comienzo de la pila (tope) → **push**
- **desapilar(pila):** quita el primer elemento en el tope de la pila, sin devolverlo → **pop**
- **tope(pila):** devuelve el primer elemento de la pila, sin extraerlo → **cima, top**
- **esPilaVacía(pila):** comprueba si la pila está vacía



# Aplicaciones de pilas

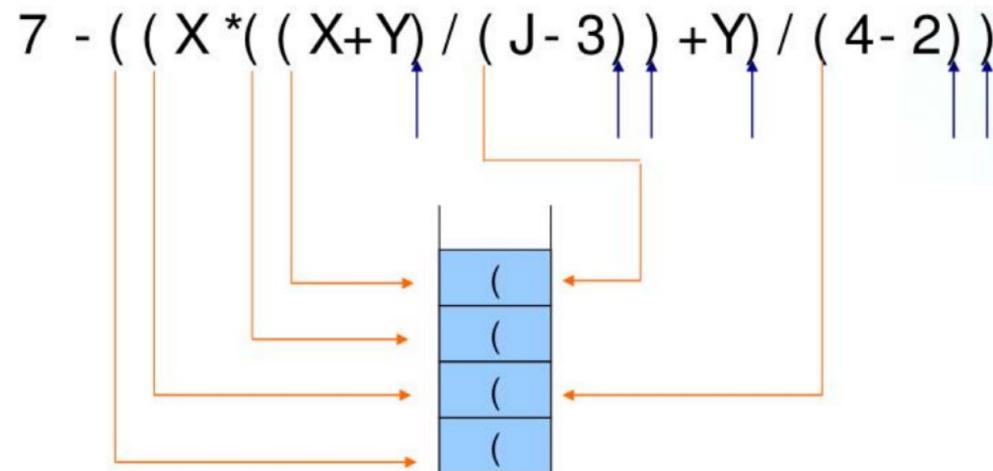
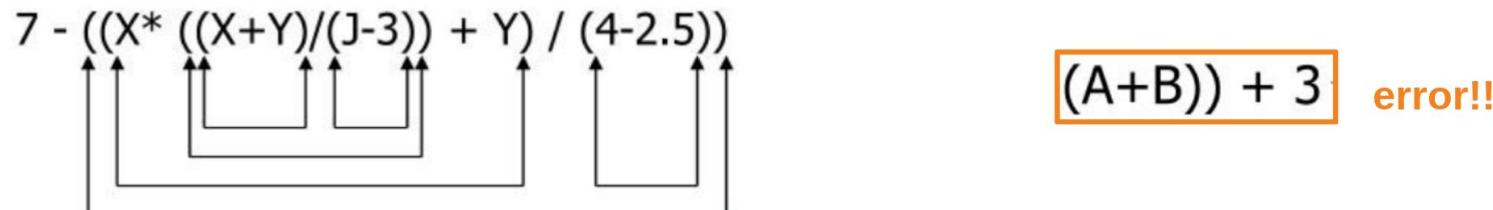
- **Ejemplo 1:** Cambio de base





# Aplicaciones de pilas

- **Ejemplo 2:** Verificación de paréntesis





# Aplicaciones de pilas

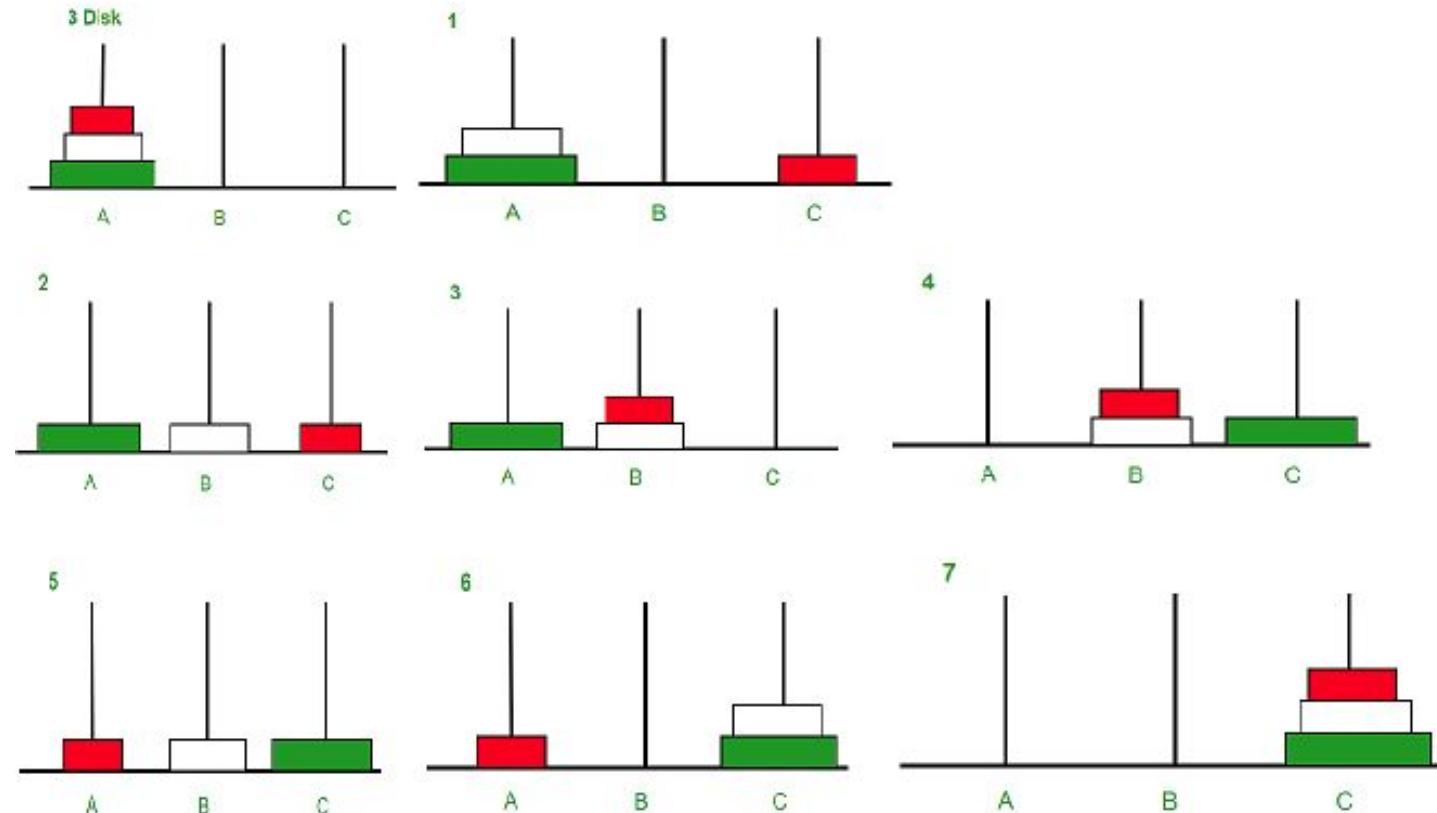
## • Ejemplo3: Torres de Hanoi

### Objetivo:

Cambiar pila de soporte A a C

### Reglas:

- se mueve un disco a la vez
- pilas en soportes siempre están ordenadas según tamaño de discos (los más grandes abajo)

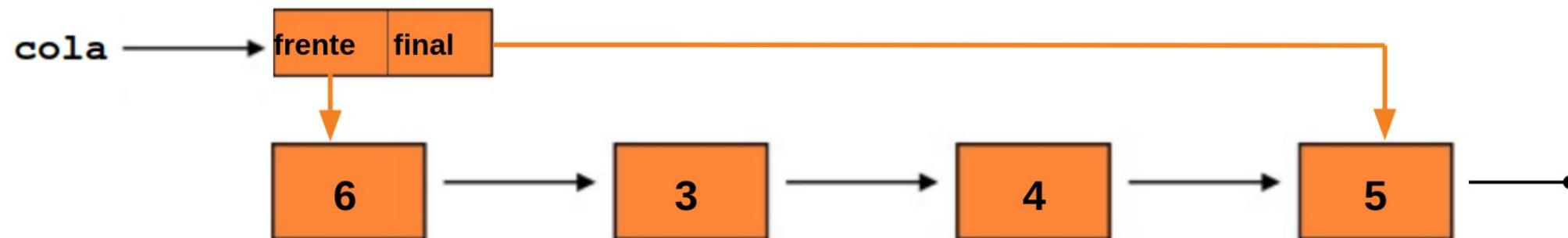
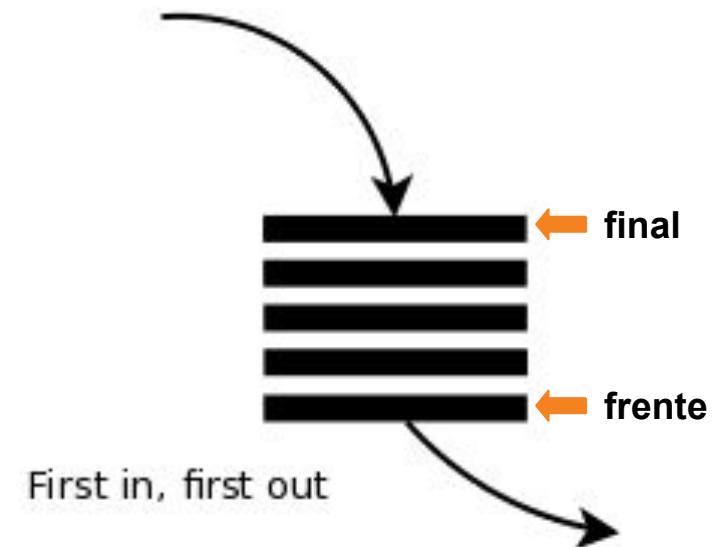




# Especificación de TDA cola

- **Estructura de datos:**

- Una cola es una secuencia de elementos que cumple la propiedad **FIFO** (first in, first out), es decir, el primer elemento que entra a la cola, será el primer elemento en salir de ella





# Especificación de TDA cola

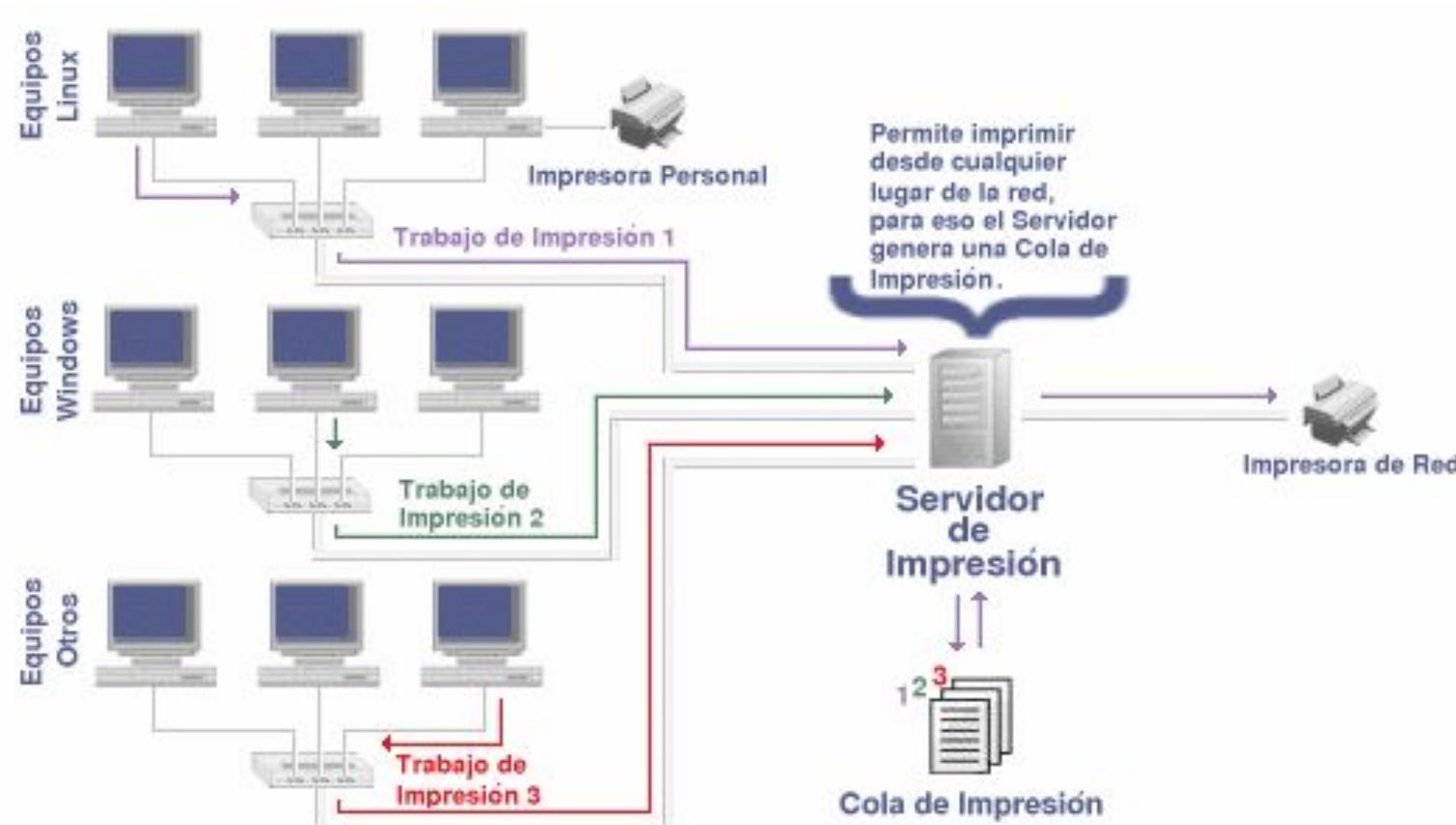
- **Operaciones:**

- **encolar(cola,dato):** agrega un elemento al final (back) de la cola → **enqueue**
- **descolar(cola):** quita el primer elemento en el frente (front) de la cola → **dequeue**
- **frente(cola):** obtiene el primer elemento de la cola, sin extraerlo → **front**
- **final(cola):** obtiene el último elemento de la cola, sin extraerlo → **back**
- **esColaVacía(cola):** comprueba si la cola está vacía



# Aplicaciones de colas

- Ejemplo: Cola de impresión





# Actividades



# Actividad en conjunto - 5+10 minutos

- **Ejercicio 1:** Escribir un algoritmo para mostrar una secuencia de enteros en orden inverso al ingresado usando TDA pila

Ejemplo:

ENTRADA: 8 6 5 7 3 9

SALIDA: 9 3 7 5 6 8





# Actividad en conjunto - 5+10 minutos

- **Ejercicio 1:** Escribir un algoritmo para mostrar una secuencia de enteros en orden inverso al ingresado usando TDA pila

Ejemplo:

ENTRADA: 8 6 5 7 3 9

SALIDA: 9 3 7 5 6 8

```
invertirSecuencia(secuencia)
    largo ← obtenerTamaño(secuencia)
    pila ← crearPilaVacia(largo)
    para i←1 hasta largo
        apilar(pila, secuencia(i))
    para i←1 hasta largo
        aux←(tope(pila))→dato
        escribir(aux)
        desapilar(pila)
```



# Actividad en conjunto - 5+10 minutos

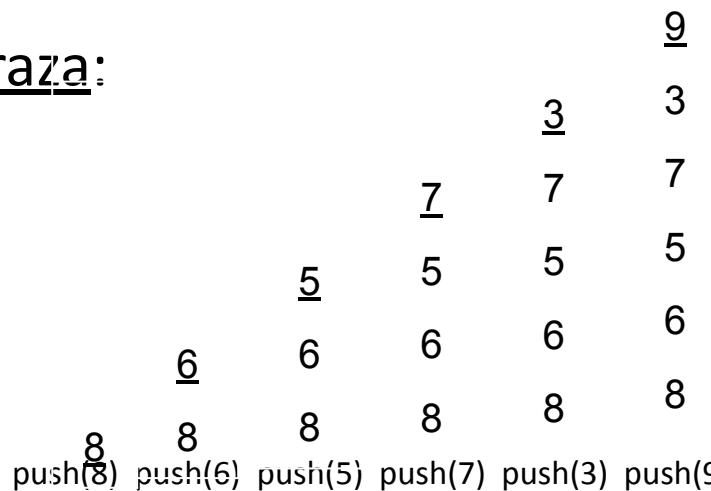
- **Ejercicio 1:** Escribir un algoritmo para mostrar una secuencia de enteros en orden inverso al ingresado usando TDA pila

Ejemplo:

ENTRADA: 8 6 5 7 3 9

SALIDA: 9 3 7 5 6 8

Traza:



```
invertirSecuencia(secuencia)
    largo ← obtenerTamaño(secuencia)
    pila ← crearPilaVacia(largo)
    para i←1 hasta largo
        apilar(pila,secuencia(i))
    para i←1 hasta largo
        aux←(tope(pila))→dato
        escribir(aux)
        desapilar(pila)
```



# Actividad en conjunto - 5+10 minutos

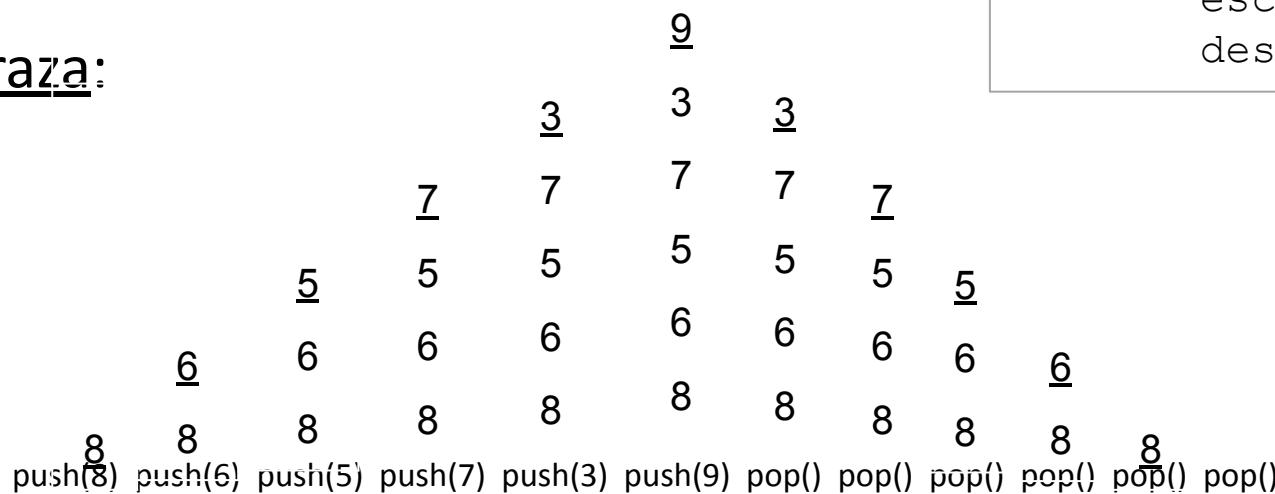
- **Ejercicio 1:** Escribir un algoritmo para mostrar una secuencia de enteros en orden inverso al ingresado usando TDA pila

Ejemplo:

ENTRADA: 8 6 5 7 3 9

SALIDA: 9 3 7 5 6 8

Traza:



```
invertirSecuencia(secuencia)
largo ← obtenerTamaño(secuencia)
pila ← crearPilaVacia(largo)
para i←1 hasta largo
    apilar(pila,secuencia(i))
para i←1 hasta largo
    aux←(tope(pila))→dato
    escribir(aux)
    desapilar(pila)
```

Salida:  
9 3 7 5 6 8

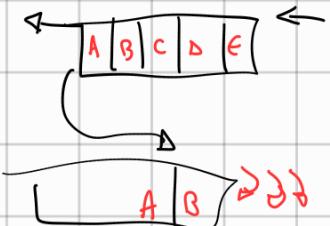


# Actividad en parejas - 15 minutos

- **Ejercicio 2:** Escribir un algoritmo en pseudocódigo que reciba una cola Q de enteros y dos valores v1 y v2, y que devuelva la cola  $Q'$  luego de reemplazar todas las ocurrencias de v1 por  $(v2-v1)$  en la cola Q. La solución puede usar colas auxiliares pero ningún otro TDA. Calcular y justificar la complejidad del algoritmo propuesto.

la misma  
cola  
cambia

+ idea



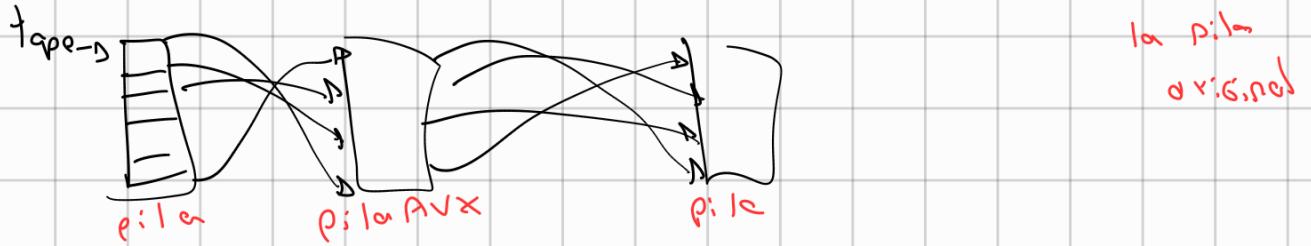
algoritmo ( $\text{cola}, \text{num } V_1, \text{num } V_2$ ):  $\text{cola}$

$O(1)$  {       $\text{Salida} \leftarrow \text{CrearColaVacia}()$   
MIENTRAS  $\text{No}(\text{esColaVacia}(\text{cola}))$  HACER  
     $\text{datoFrente} \leftarrow (\text{tope}(\text{cola}) \rightarrow \text{dato})$   
     $\text{descolar}(\text{cola})$   
    SI  $\text{datoFrente} = V_1$  ENTONCES  
         $\text{encolar}(\text{salida}, V_2 - V_1)$   
    SINO  
         $\text{encolar}(\text{salida}, \text{datoFrente})$   
 $O(1)$  {      DESENGUER  $\text{Salida}$



# Actividad en parejas - 15 minutos

- **Ejercicio 3:** Escribir un algoritmo en seudocódigo que reciba una pila  $S$  de enteros y un valor  $v$ , y que devuelva 1 si el valor  $v$  se encuentra en  $S$ , y 0 en caso contrario. Al terminar el algoritmo, la pila  $S$  debe quedar exactamente igual a como ingresó. La solución puede usar pilas auxiliares pero ningún otro TDA. Calcular y justificar la complejidad del algoritmo propuesto.



Algoritmo ( $pila\ S$ , num  $v$ ): num

$pila\ Aux \leftarrow crearPilaVacia()$

resultado  $\leftarrow 0$

MIENTRAS  $No(esPilaVacia(S))$  HACER

$tope \leftarrow tope(S) \rightarrow dato$

Si  $Tope = V$  ENTONCES

resultado  $\leftarrow 1$

desapilar( $S$ )

apilar( $pila\ Aux$ ,  $tope$ )

MIENTRAS  $No(esPilaVacia(pilaAux))$  HACER

$topeAux \leftarrow tope(pilaAux) \rightarrow dato$

desapilar( $pila\ Aux$ )

apilar( $S$ )

DESVOLVER resultado



# Actividad asíncrona

- **Ejercicio 4:** Escribir un algoritmo en seudocódigo que reciba una pila  $S$  y devuelva una pila  $S'$  con todos los elementos de  $S$  ordenados ascendente, con el mayor elemento en la cima y el menor elemento en el fondo. La solución puede usar pilas auxiliares pero ningún otro TDA. Calcular y justificar la complejidad del algoritmo propuesto.

Menor



Menor

4  
 1  
 9  
 10  
 5

IDEA "Recorro pila buscando el número menor,

Luego lo apilo en una nueva pila

algor:TMO (pila S) : pila

auxPila  $\leftarrow$  crearPilaVacia() debería tener la capacidad

PilaFinal  $\leftarrow$  crearPilaVacia()

$n \leq 0$

MIENTRAS  $\neg(\text{esPilaVacia}(S))$  HACER

top  $\leftarrow$  tope(S)  $\Rightarrow$  dato

desapilar(S)

apilar(auxPila, top)

$n \leq n + 1$

} contener elementos Pila

MIENTRAS  $\neg(\text{esPilaVacia}(\text{auxPila}))$  HACER

top  $\leftarrow$  tope(auxPila)  $\Rightarrow$  dato

desapilar(auxPila)

apilar(S, top)

} pila S vuelve al estado original

PARA  $i \leq 1$  HASTA n

Menor  $\leftarrow$  tope(S)  $\Rightarrow$  dato ... Para la 1era iteración se toma como menor el 1er num.

MIENTRAS  $\neg(\text{esPilaVacia}(S))$  HACER

numActual  $\leftarrow$  tope(S)  $\Rightarrow$  dato

Sí numActual  $\leq$  Menor Entonces

Menor  $\leftarrow$  tope(S)  $\Rightarrow$  dato

desapilar(S)

apilar(auxPila, numActual)

} buscamos valor menor en la pila

MIENTRAS  $\neg(\text{esPilaVacia}(\text{auxPila}))$  HACER

numActual  $\leftarrow$  tope(auxPila)  $\Rightarrow$  dato

desapilar(auxPila)

Sí numActual  $>$  Menor

apilar(S, menor) ... no se apila el menor de nuevo en la Pila S

} copiar n pilas

apilar(PilaFinal, menor) ... agregamos menor en pila final.

DE VOLVER PilaFinal



# Actividad asíncrona

- **Ejercicio 5:** Dada una secuencia de dígitos del 0 al 9, representada como una lista simplemente enlazada, escribir un algoritmo en pseudocódigo que elimine los dígitos repetidos, tal que cada dígito aparezca una única vez. El algoritmo debe mostrar una lista simplemente enlazada que representa el menor número posible entre todos los posibles resultados. Por ejemplo, si la secuencia de entrada es 23123, la secuencia de salida debería ser 123. Solo se pueden ocupar los **TDAs lista simplemente enlazada y pila**. Calcular y justificar la complejidad del algoritmo propuesto.

Idea: parecido al anterior, recorro la LF buscando el dato menor, obrelo el dato a otras LF y borro todos los nodos que tengan el dato de la LF original. Vuelvo a hacer lo mismo hasta que la LF sea vacía.

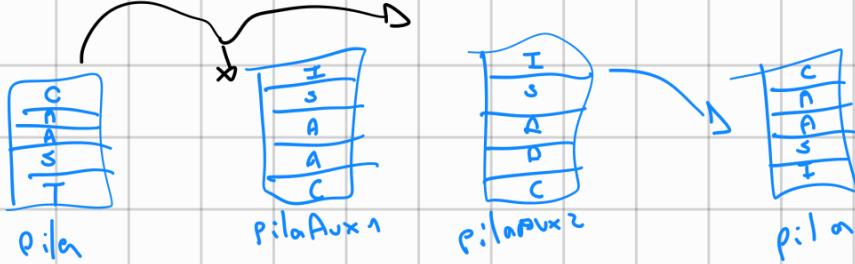


## Incentivo 4 - 15+5 minutos

- Escribir un algoritmo para que dada una palabra, indique si es o no palíndroma (que se lee igual de derecha a izquierda y de izquierda a derecha). Solo se puede usar TDA pila ~~o~~ TDA cola para resolver este ejercicio. ¿Cuál es el orden de complejidad del algoritmo propuesto?

Idea

Version 1



E<sub>s</sub> Palíndromo (LG palabra): booleano  
auxPila <→ pila vector  
pila <→ CrearPilaVacia()

M: ENTRAS auxPila < $\neq$  NULO HACER

apilar(pila, auxPila → dato)

auxPila <→ auxPila → puntero

pila Aux1 <→ CrearPilaVacia()

pila Aux2 <→ CrearPilaVacia()

M: ENTRAS NO(esPilaVacia(pila)) HACER

top <→ Tope(pila) → dato

desapilar(pila)

apilar(pilaAux1, top)

apilar(pilaAux2, top)

M: ENTRAS NO(esPilaVacia(pilaAux1)) HACER

top <→ Tope(pilaAux1) → dato

desapilar(pilaAux1)

apilar(pila)

... Comprobar palíndromo

M: ENTRAS NO(esPilaVacia(pila)) HACER

si Tope(pila) → dato < $\neq$  Tope(pilaAux2) → dato ENTONCES

DE VOLVER FALSO

desapilar(pila)

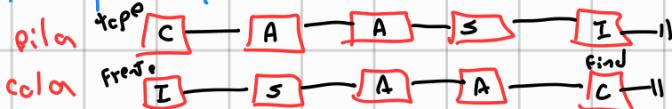
desapilar(pilaAux2)

DE VOLVER VERDADEIRO

\* Solo uso TDA PILA

IDEA. Creo pila y cola con la palabra, luego voy desrolando y desapilando comprobando que las letras NO sean diferentes

Versión 2



esPalindromo (LE palabra) : booleano

aux < palabra

cola < CrearColaVacia()

MIENTRAS palabra <> NULO HACER

Encolar (cola, palabra  $\rightarrow$  dato)

palabra < palabra  $\rightarrow$  puntero

pila < CrearPilaVacia()

aux < palabra

MIENTRAS palabra <> NULO HACER

Apilar (pila, palabra  $\rightarrow$  dato)

palabra < palabra  $\rightarrow$  puntero

... comparar palabra

MIENTRAS No(esPilaVacia(pila)) HACER

S: Top(pila)  $\rightarrow$  dato <> Frente(cola)  $\rightarrow$  dato

DE VOLVER FALSO

desapilar (pila)

descolar (cola)

DE VOLVER VERDADERO

USÓ TDA pila y TDA Cola



# Incentivo 4 - 15+5 minutos

## Puntaje

I.	El algoritmo propuesto apunta a resolver el problema planteado. (SI 1 punto / NO 0 punto)	
II.	El algoritmo resuelve correctamente el problema planteado. (SI 1 punto / NO 0 punto)	
III.	El algoritmo está escrito en pseudocódigo ordenado (SI 1 punto / NO 0 punto)	
IV.	El algoritmo está escrito en el formato establecido (SI 1 punto / NO 0 punto)	
V.	El algoritmo identifica entradas correctamente (SI 1 punto / NO 0 punto)	
VI.	El algoritmo identifica y declara salidas de manera correcta. (SI 1 punto / NO 0 punto)	
VII.	Calcula correctamente la complejidad (SI 1 punto / NO 0 punto)	
VIII.	Justifica la complejidad del algoritmo propuesto. (SI 1 punto / NO 0 punto)	
	PUNTOS	



# Actividad de cierre



- Ir a [menti.com](https://www.menti.com) e ingresar código 3209 0457



# Próximas fechas...

- Resumen de la semana:
  - TDA pila
  - TDA cola

~~cátedra - refuerzo~~ – laboratorio

U2 - S7

Octubre 2023						
Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

- Próxima semana:
  - TDA grafo

Noviembre 2023						
Receso						
Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		