DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Análisis de Algoritmos y Estructura de Datos

TDA pila y TDA cola

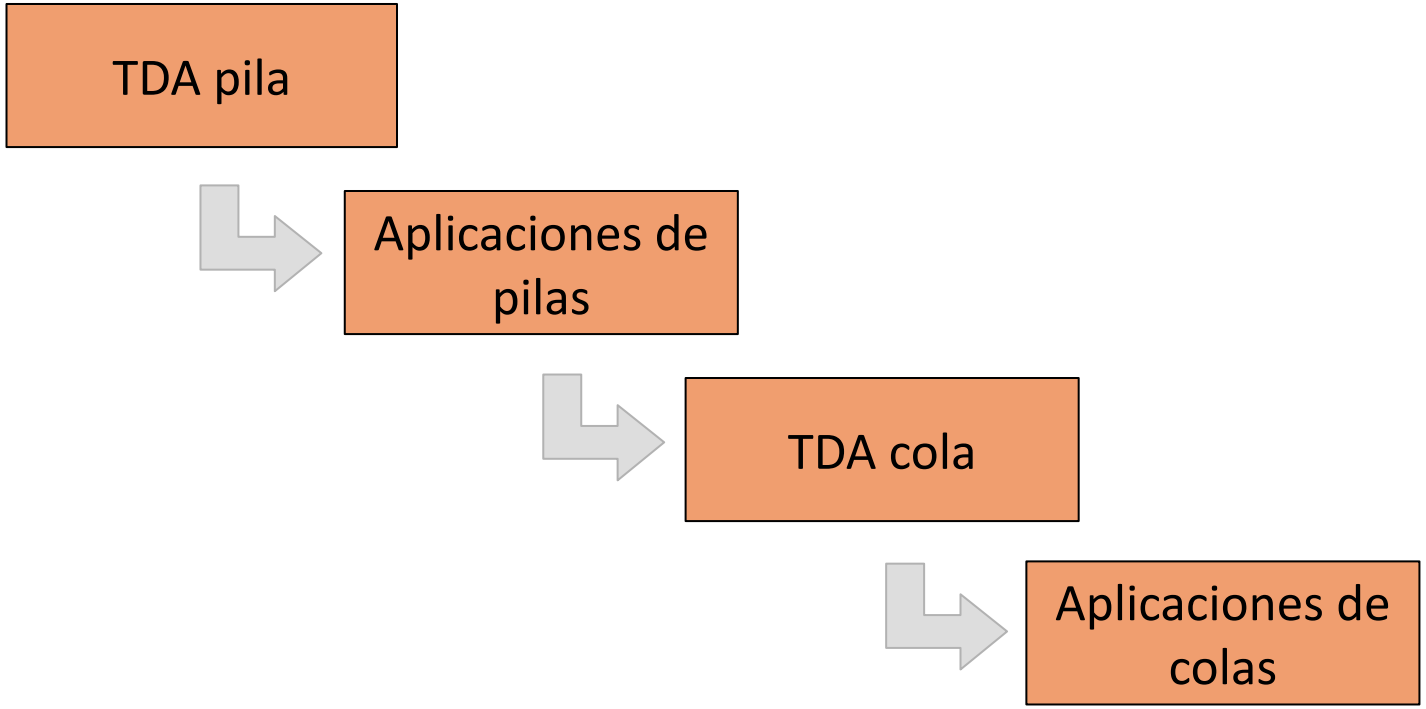
Prof. Violeta Chang C

Semestre 2 – 2023

TDA Pila y TDA Cola

- **Contenidos**:
  - Especificación formal de TDA pila
  - Operaciones/aplicaciones con pilas
  - Especificación formal de TDA cola
  - Operaciones/aplicaciones con colas
- **Objetivos**:
  - Entender y explicar estructura de datos de TDA pila
  - Comprender funcionamiento de operaciones con pilas y determinar su complejidad
  - Conocer aplicaciones clásicas de pilas
  - Entender y explicar estructura de datos de TDA cola
  - Comprender funcionamiento de operaciones con colas y determinar su complejidad
  - Conocer aplicaciones clásicas de colas

Ruta de la sesión



Especificación de TDA pila

- **Idea general:**
  - Imaginemos una torre de platos: colocamos los platos uno encima del otro

- Para evitar que los platos se caigan, es preciso:
  - agregar siempre un plato nuevo encima de la torre de platos y no en otra ubicación
  - retirar siempre el plato que está encima de la torre de platos y no desde otro lugar de la torre

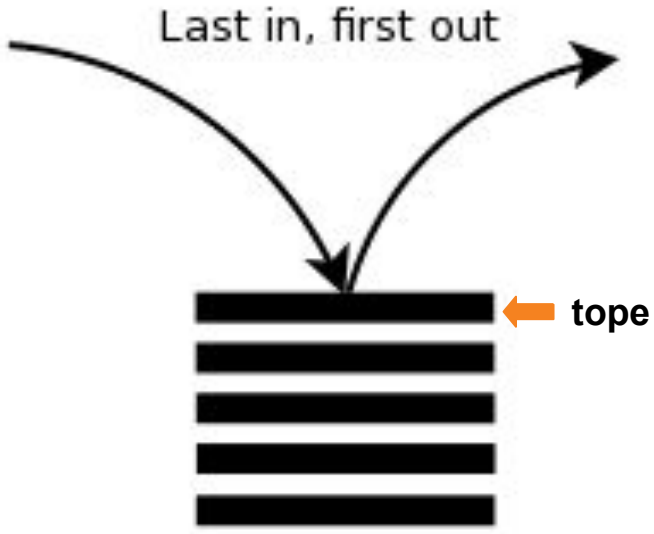
Especificación de TDA pila

- **Estructura de datos**:
  - Una pila (o stack) es una colección lineal con componentes homogéneos y capacidad fija
  - Homogéneo: Todos los componentes son del mismo tipo
  - Lineal: Componentes están ordenados en una línea

  - Capacidad fija: Desbordamiento de pila

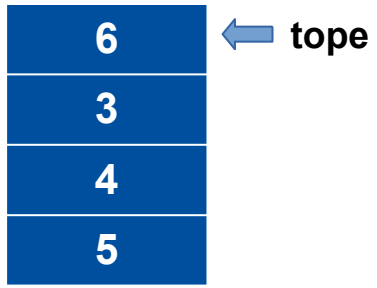
Especificación de TDA pila

- **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
  - **Tope de la pila**: extremo de la lista por donde se introducen y extraen los elementos



Especificación de TDA pila

- **Estructura de datos**: **<u>Ejemplo</u>**
  - Edades es una instancia del TDA pila y almacena las edades en años de ciertos niños en un jardín de infancia:
    - Todos los elementos de la pila deben ser del mismo tipo (enteros)
    - El elemento de más arriba es el elemento de la cima, por lo tanto los elementos fueron ingresados en el siguiente orden : 5, 4, 3, 6.
    - El elemento 3 no puede eliminarse de la pila mientras el elemento 6 no sea retirado. Sólo podrán ser retirados en el orden siguiente : 6, 3, 4, 5.



Edades

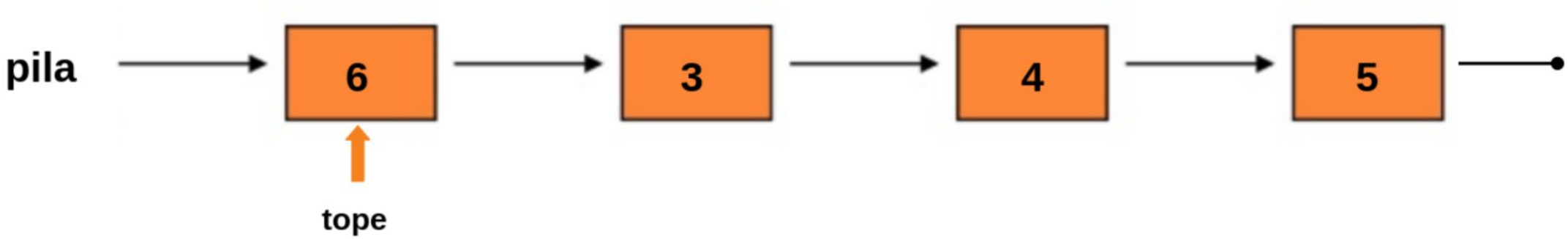
Especificación de TDA pila

- **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

Representación de TDA pila

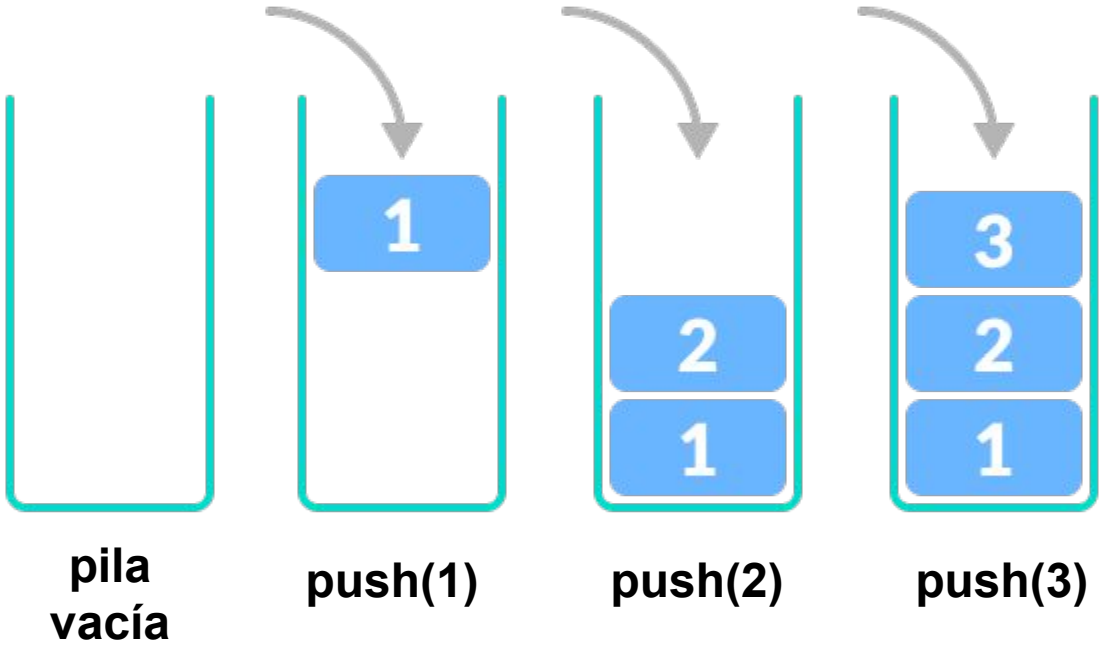
- **Dos opciones:**
  - Usando un arreglo:

  - **Usando una lista enlazada simple:**



Operación APILAR

- **Ejemplo:**
  - Supongamos que tenemos una pila de enteros vacía y queremos introducir los valores 1, 2 y 3:



Operación APILAR

- La operación *apilar* (PUSH) permite agregar un elemento a una pila, siempre por el tope

```
apilar(pila,valor)
        nodo ← crearNodoVacio()
        nodo→dato ← valor
        nodo→puntero ← pila
        pila ← nodo
```

Operación APILAR

- La operación *apilar* (PUSH) permite agregar un elemento a una pila, siempre por el tope

```
apilar(pila,valor)
        nodo ← crearNodoVacio()
        nodo→dato ← valor
        nodo→puntero ← pila
        pila ← nodo
```

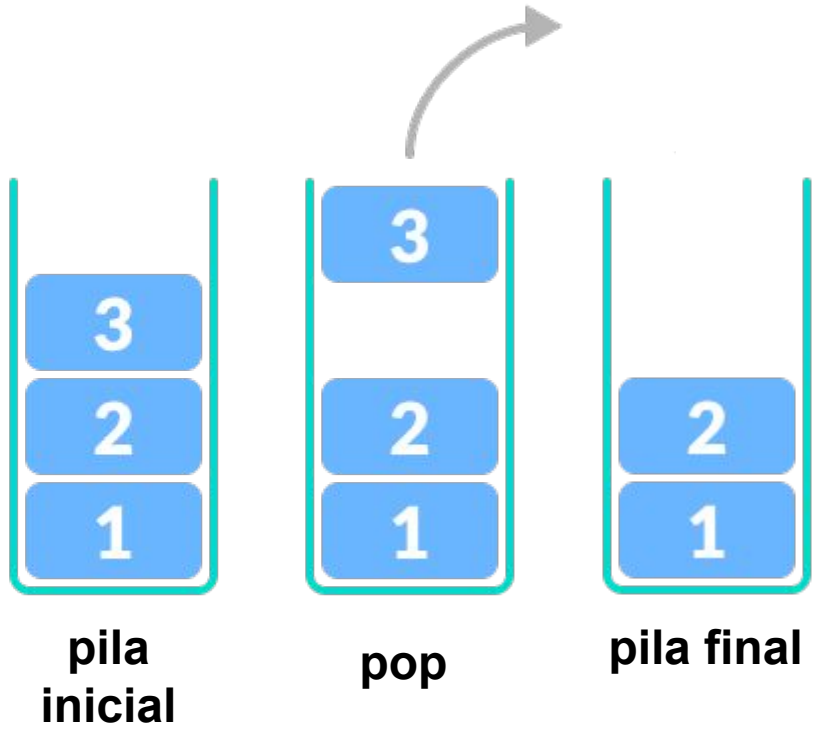
O(1)

Orden de complejidad: O(1)

Similar a inserción al inicio de lista enlazada simple

Operación DESAPILAR

- **Ejemplo:**
  - Supongamos que tenemos una pila de enteros con los valores 1,2 y 3, siendo 3 el elemento del tope de la pila. Necesitamos extraer el elemento del tope:



Operación DESAPILAR

- La operación *desapilar* (POP) permite extraer el elemento del tope de una pila, pero no devuelve dicho elemento

```
desapilar(pila)
        nodo ← pila
        pila ← pila→puntero
        liberar(nodo)
```

Operación DESAPILAR

- La operación *desapilar* (POP) permite extraer el elemento del tope de una pila, pero no devuelve dicho elemento

```
desapilar(pila)
        nodo ← pila
        pila ← pila→puntero
        liberar(nodo)
```

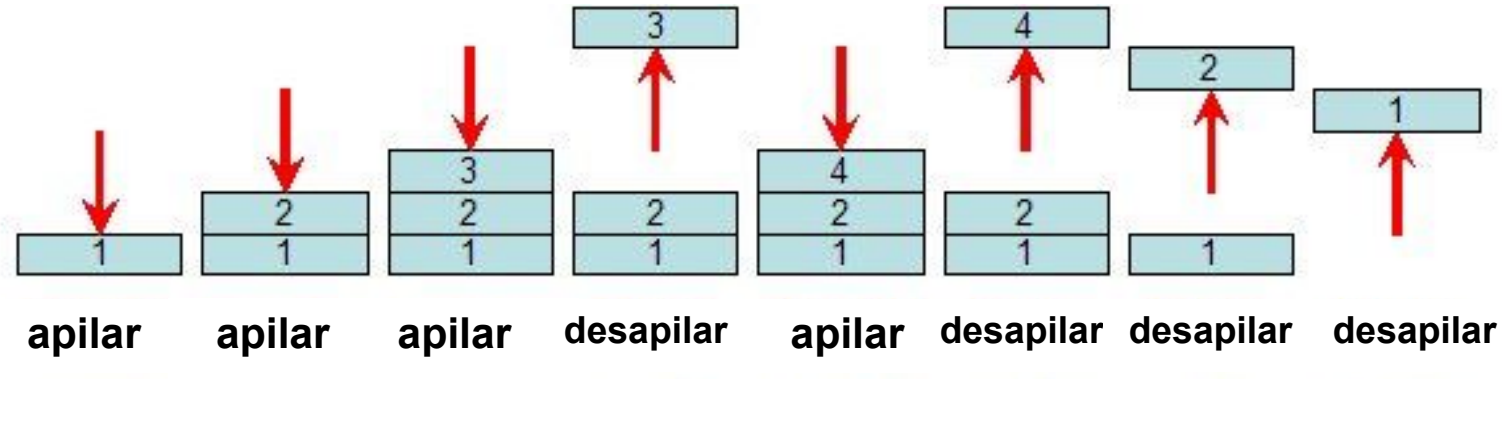
O(1)

Orden de complejidad: O(1)

Similar a eliminación al inicio de lista enlazada simple

Operaciones APILAR y DESAPILAR

- **Ejemplo:**
  - Supongamos que tenemos una pila de enteros, inicialmente vacía, que contiene valores sucesivos, comenzando de 1.
  - Una secuencia de operaciones push y pop dejaría la pila vacía nuevamente:



Ejercicio propuesto

- **<u>Ejercicio</u>:** Escribir un algoritmo para que dada una pila, devuelva la cantidad de elementos de dicha pila. La pila ingresada debe mantenerse idéntica después de finalizar el algoritmo. ¿Cuál es el orden de complejidad del algoritmo propuesto?

Aplicaciones de pilas

- Las aplicaciones más comunes de las pilas son:
  - Inversión de cadenas
  - Detección de palíndromas
  - Verificación de parentización
  - Análisis sintáctico
  - Conversión de notaciones
  - Cambio de base

Aplicaciones de pilas

- Llamadas a subprogramas

```
function one()
{
   int a = 10;
   print(a);
}
```

In-depth introduction to call stack, @Medium

Aplicaciones de pilas

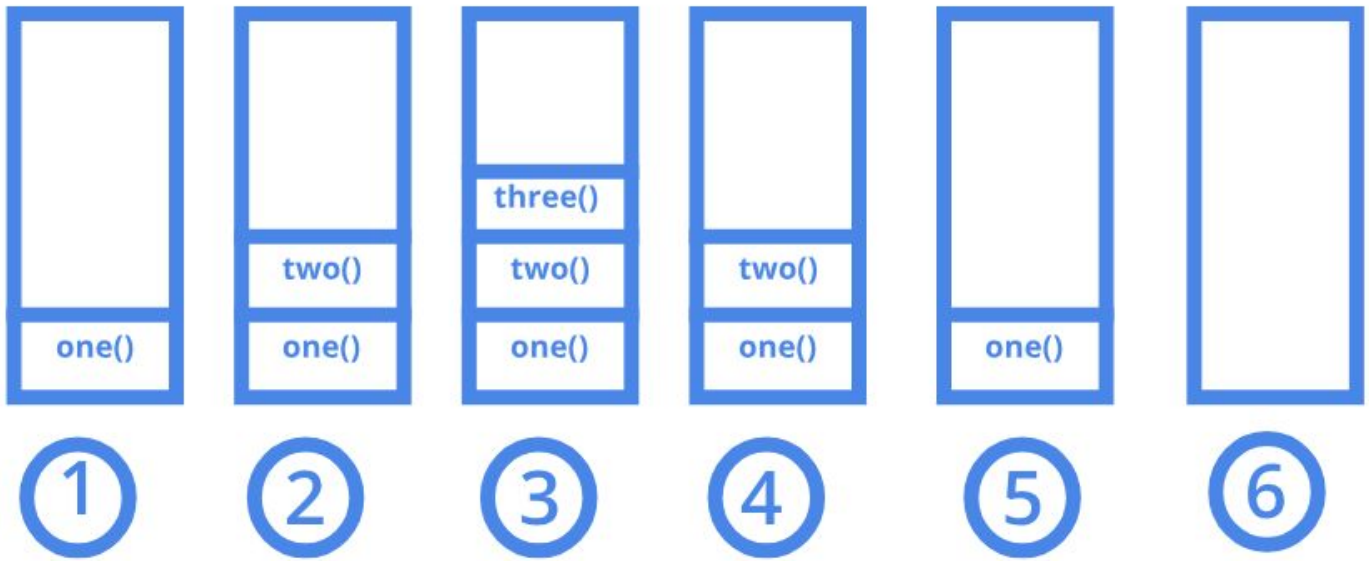
- Llamadas a subprogramas

```
function one()
{
 two();
}

function two()
{
 three();
}

function three()
{
 mostrar("Pila de llamadas");
}

main()
{
 one();
}
```



In-depth introduction to call stack, @Medium

Especificación de TDA cola

- **Idea general:**
  - Imaginemos una fila de personas para pagar en una librería: las personas se ubican una detrás de otra

- La primera persona que llega a la cola será la primera en ser atendida
- Si llega otra enseguida, se ubica a continuación
- Así, sucesivamente hasta que uno a uno van siendo atendidos

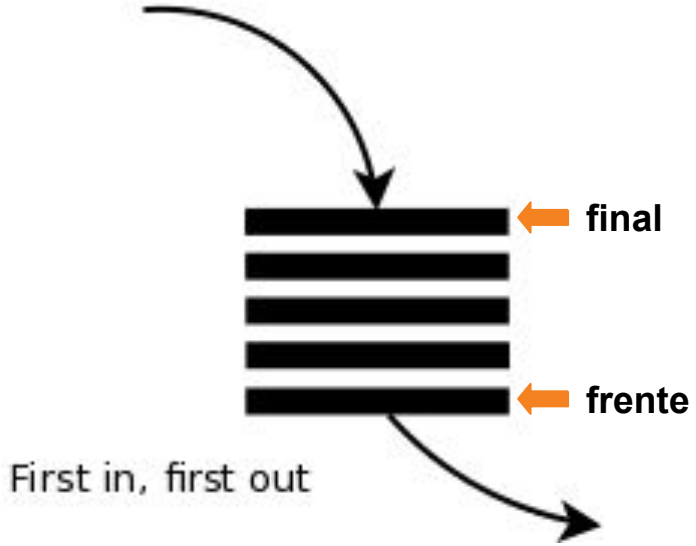
Especificación de TDA cola

- **Estructura de datos**:
  - Una cola (o queue) es una colección lineal con componentes homogéneos y capacidad fija
  - Homogéneo: Todos los componentes son del mismo tipo
  - Lineal: Componentes están ordenados en una línea

  - Capacidad fija: Desbordamiento de cola

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
  - **Frente de la cola**: extremo de la cola por donde se extraen los elementos
  - **Final de la cola**: extremo de la cola por donde se introducen los elementos



Especificación de TDA cola

- **Estructura de datos**: **Ejemplo**
  - *Operaciones* es una instancia del TAD cola que almacena las operaciones aritméticas que deben realizarse con una secuencia de números reales:

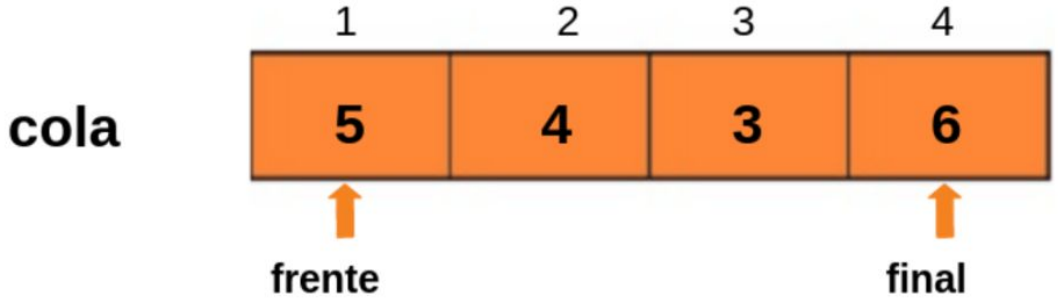
  - Todos los elementos de la cola deben ser del mismo tipo (caracteres)
  - El elemento de más a la izquierda es el elemento del frente, por lo tanto, ha sido el primero en ingresar a la lista. Según esto, debe ser el primero en abandonar la cola; los elementos deben abandonar la cola en el orden siguiente : S, R, M, D, R, M, R.
  - El elemento de más a la derecha es el elemento del fondo de la cola (el último en ingresar), por lo tanto los elementos fueron ingresados en el siguiente orden : S, R, M, D, R, M, R.
  - El elemento D no puede eliminarse de la cola mientras los elementos S, R y M (en ese orden) no sean retirados. No es posible acceder a otros elementos que no sean el frente (para eliminarlo) y el fondo (para insertar otro a continuación).

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

Representación de TDA cola

- **Dos opciones:**
  - Usando un arreglo:

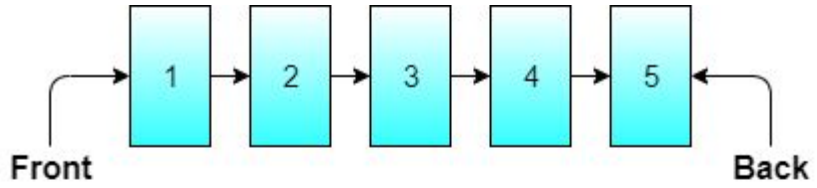


  - **Usando una lista enlazada con cabecera:**

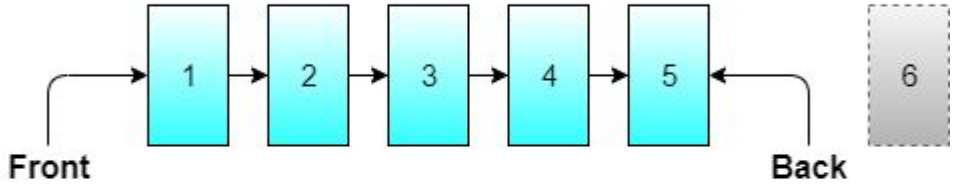
Operación ENCOLAR

- **Ejemplo:**
  - Supongamos que tenemos una cola de enteros y queremos introducir el valor 6:

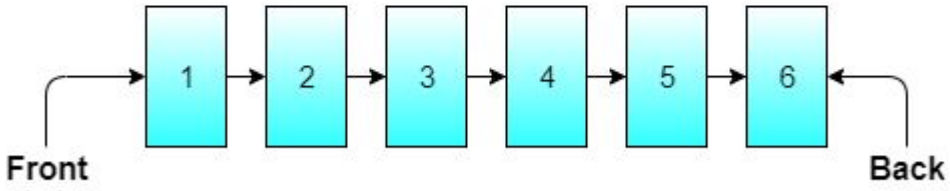
Cola original



Nuevo elemento: 6



Cola final



AfterAcademy: Queues and its basic operations

Operación ENCOLAR

- La operación *encolar* (ENQUEUE) permite agregar un elemento a una cola, siempre por el final

```
encolar(cola,valor)
        nodo ← crearNodoVacio()
        nodo→dato ← valor
        nodo→puntero ← NULO
        (cola→final)→puntero ← nodo
        cola→final ← nodo
```

Operación ENCOLAR

- La operación *encolar* (ENQUEUE) permite agregar un elemento a una cola, siempre por el final

```
encolar(cola,valor)
        nodo ← crearNodoVacio()
        nodo→dato ← valor
        nodo→puntero ← NULO
        (cola→final)→puntero ← nodo
        cola→final ← nodo
```

O(1)

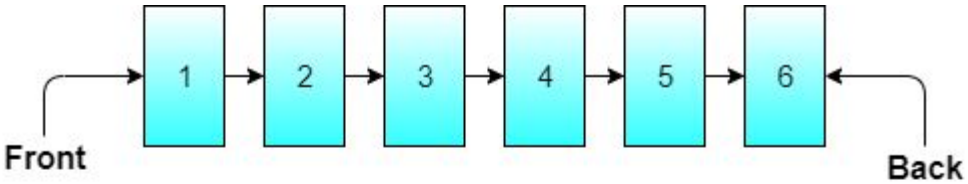
Orden de complejidad: O(1)

Similar a inserción al inicio de lista enlazada

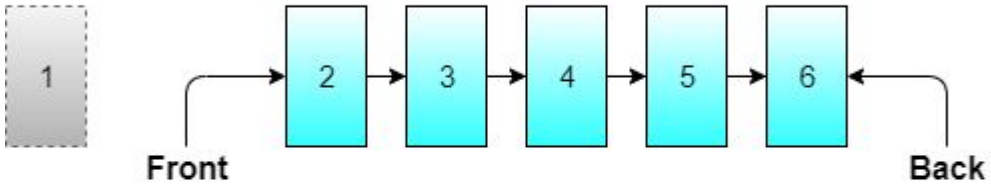
Operación DESCOLAR

- **Ejemplo:**
  - Supongamos que tenemos una cola de enteros y queremos remover un elemento::

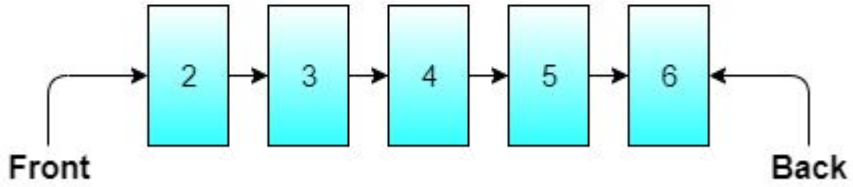
Cola original



Descolar



Cola final



AfterAcademy: Queues and its basic operations

Operación DESCOLAR

- La operación *descolar* (DEQUEUE) permite retirar un elemento de una cola, siempre por el frente

```
descolar(cola)
        nodo ← cola→frente
        cola→frente ← (cola→frente)→puntero
        liberar(nodo)
```

Operación DESCOLAR

- La operación *descolar* (DEQUEUE) permite retirar un elemento de una cola, siempre por el frente

```
descolar(cola)
        nodo ← cola→frente
        cola→frente ← (cola→frente)→puntero
        liberar(nodo)
```

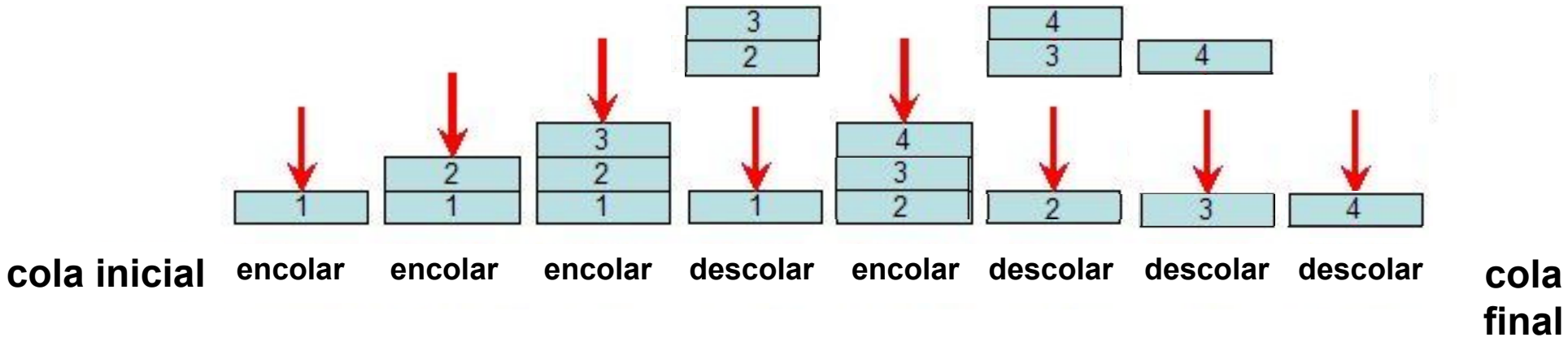
O(1)

Orden de complejidad: O(1)

Similar a eliminación al inicio de lista enlazada

Operaciones ENCOLAR y DESCOLAR

- **Ejemplo:**
  - Supongamos que tenemos una cola de enteros, inicialmente vacía, que contiene valores sucesivos, comenzando de 1.
  - Una secuencia de operaciones enqueue y dequeue dejaría la cola vacía nuevamente:



Ejercicio propuesto

- **<u>Ejercicio</u>:** Escribir un algoritmo para que dada una cola, devuelva la cantidad de elementos de dicha cola. La cola ingresada debe mantenerse idéntica después de finalizar el algoritmo. ¿Cuál es el orden de complejidad del algoritmo propuesto?

Aplicaciones de colas

- Las aplicaciones más comunes de las colas son en el contexto de gestión de recursos:
  - Sistemas de tiempo compartido (uso de memoria y procesador)
  - Colas de impresión
  - Simulación de situaciones reales:
    - cola de clientes para pagar permiso de circulación online
    - cola de espera para ser atendidos por ejecutivo de call center de LATAM

Actividad de cierre

- Ir a menti.com e ingresar código 7499 5569

Próximas fechas...

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

  ~~cátedra~~ – refuerzo – laboratorio

- Próxima semana:
  - TDA grafo

U2 - S7