4. Implementaciones de cola basadas en arreglos

Carlos Zerón Martínez

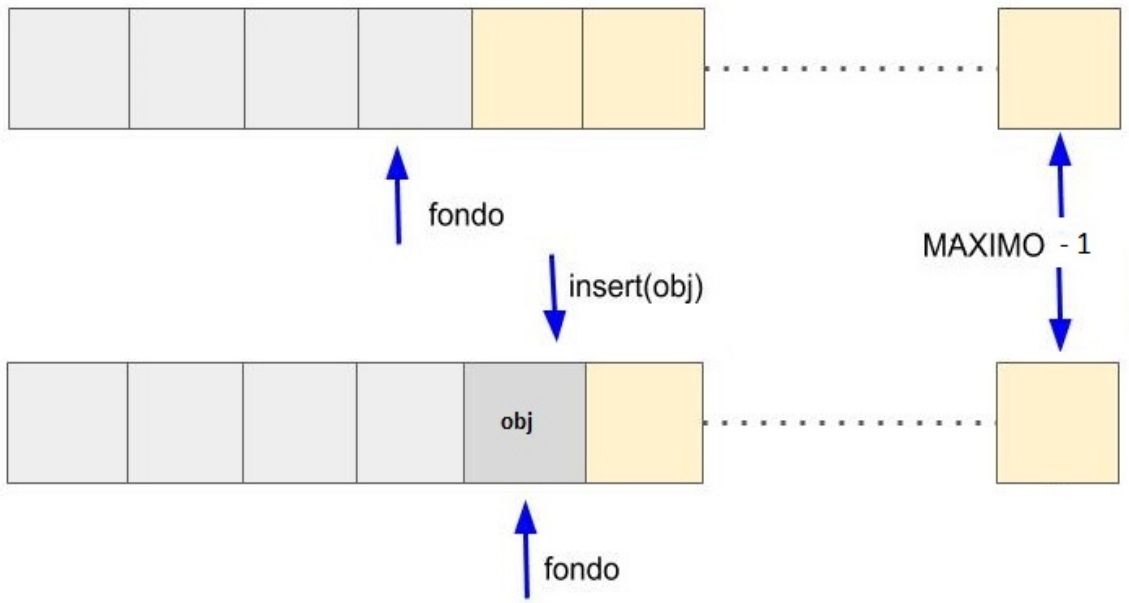
Universidad Nacional Autónoma de México

zeron@ciencias.unam.mx

Jueves 12 de Noviembre de 2020

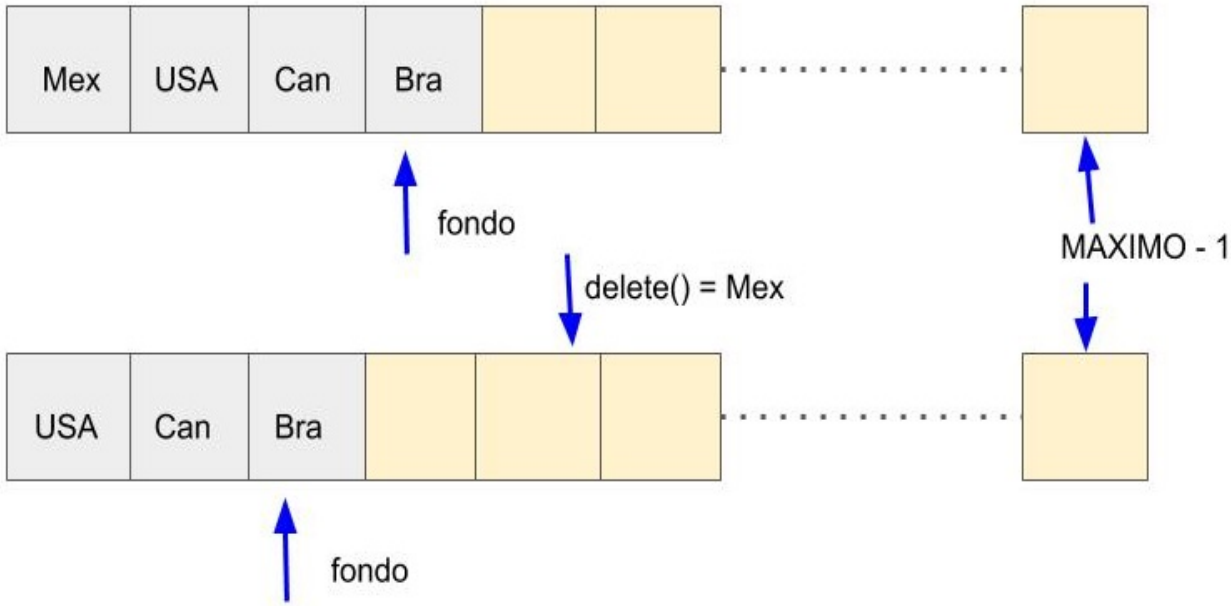
Implementación de una cola basada en un arreglo

- Además del arreglo, tendríamos una constante que indique el máximo tamaño de la cola MAXIMO y una variable que especifique el índice del fondo de la cola *fondo*. (el índice 0 representaría el frente)
- Para la inserción, se incrementa *fondo* y se almacena el nuevo objeto ahí



Implementación de una cola basada en un arreglo

- Para la eliminación, se disminuye una posición a todos los elementos que seguían al eliminado en una unidad



Problema: si hay n elementos en la cola, el tiempo de ejecución de la eliminación del elemento del frente sería $O(n)$.

Implementación de una cola basada en un arreglo circular

- Una mejor implementación considera que el elemento ubicado en la última posición del arreglo *preceda* al de la primera posición.
- *MAXIMO*: máximo de elementos a insertar
- *frente*: índice del elemento que está en el frente de la cola.
- *fondo*: índice del elemento que está al final de la cola
- *numeroElementos*: número de elementos de la cola.

Implementación de una cola basada en un arreglo circular

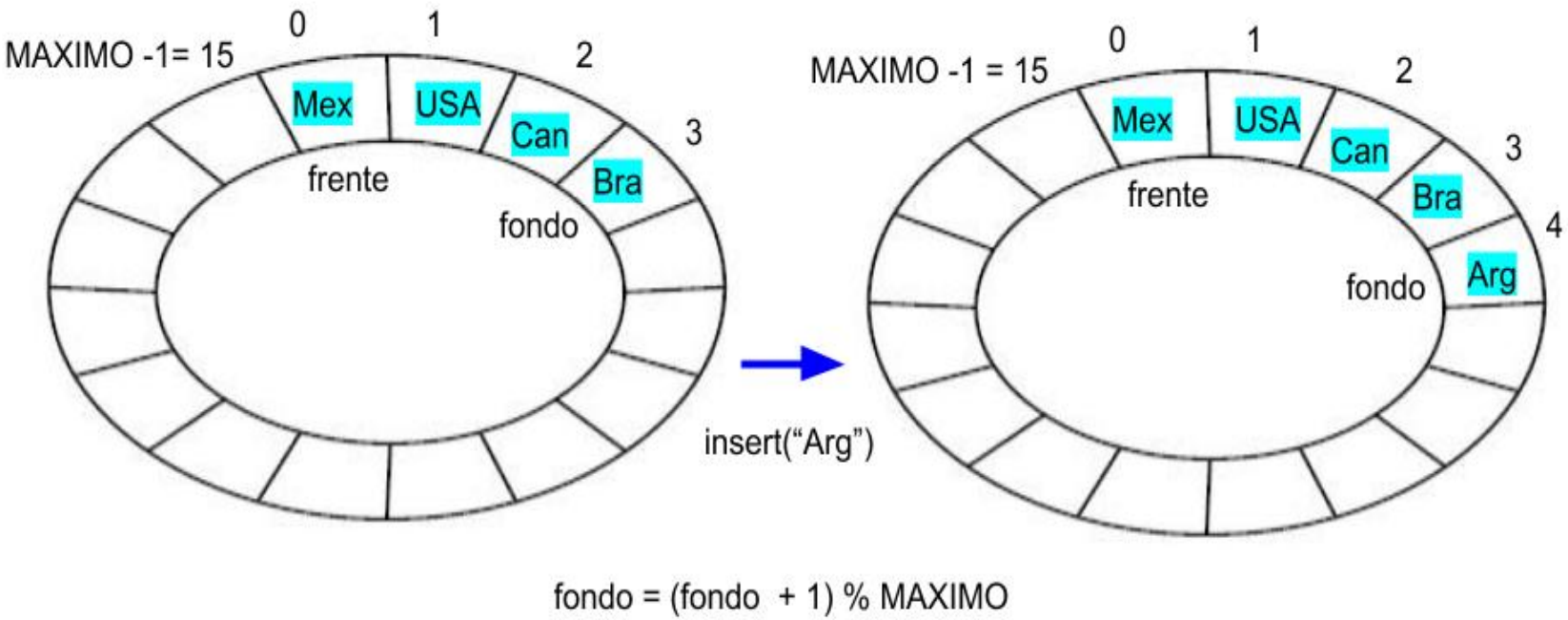
```
public class ArregloCircular {

    private final short MAXIMO = 10000;

    private Object[] cola;

    private int frente;

    private int fondo;

    private int numeroElementos;
```

Implementación de una cola basada en un arreglo circular

Inserción

En la inserción se mueve de forma circular el índice que refiere al fondo.

Ejemplo con una cola de tamaño máximo 16 con cadenas que representan países.



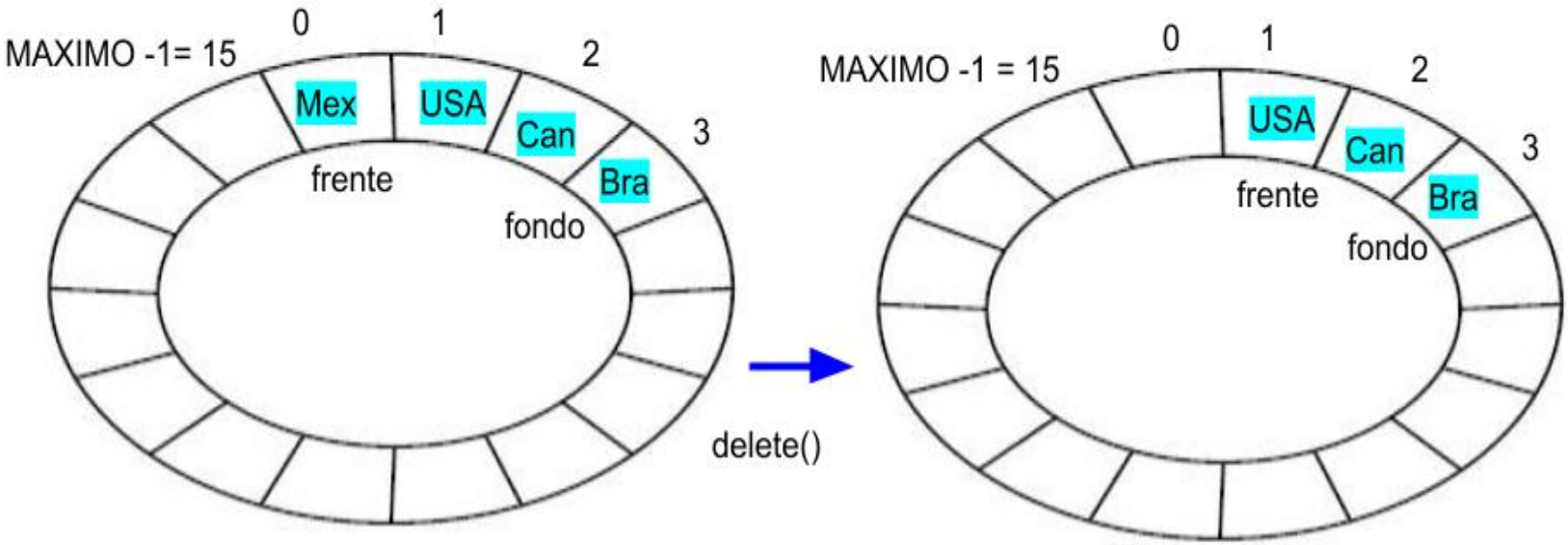
fondo = (fondo + 1) % MAXIMO

Implementación de una cola basada en un arreglo circular

Eliminación

En la eliminación se mueve de forma circular el índice que refiere al frente.

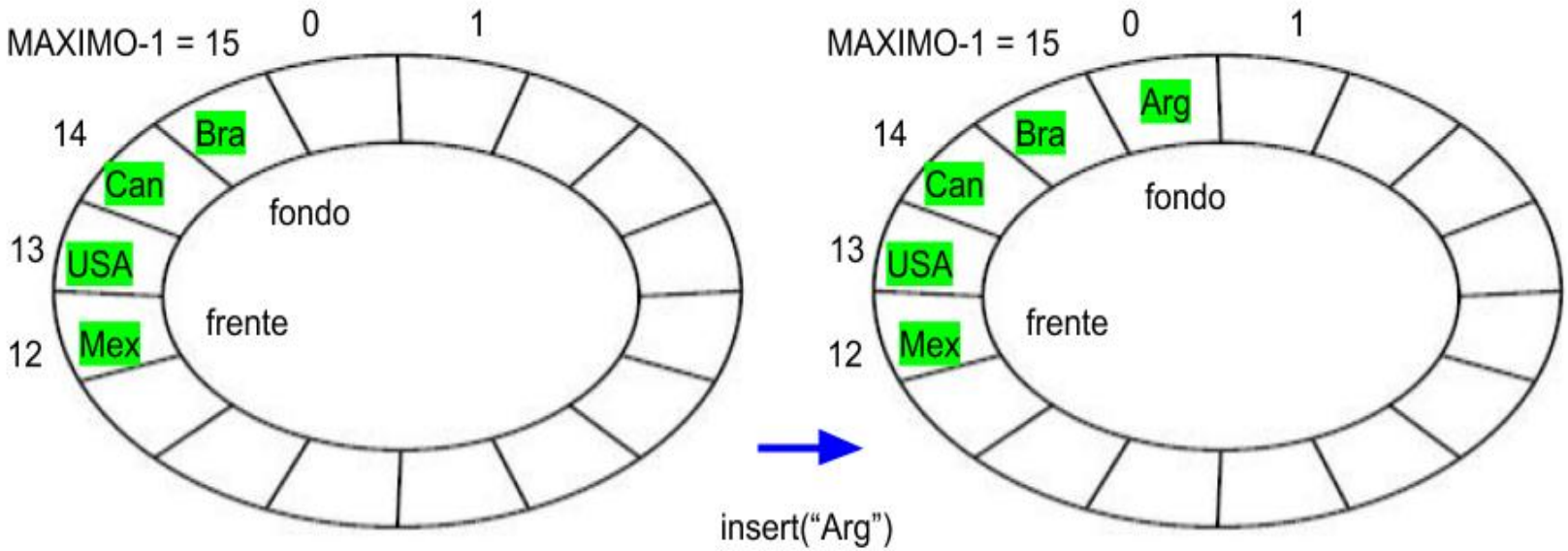
Ejemplo con una cola de tamaño máximo 16 con cadenas que representan países.



frente = (frente + 1) % MAXIMO

Implementación de una cola basada en un arreglo circular

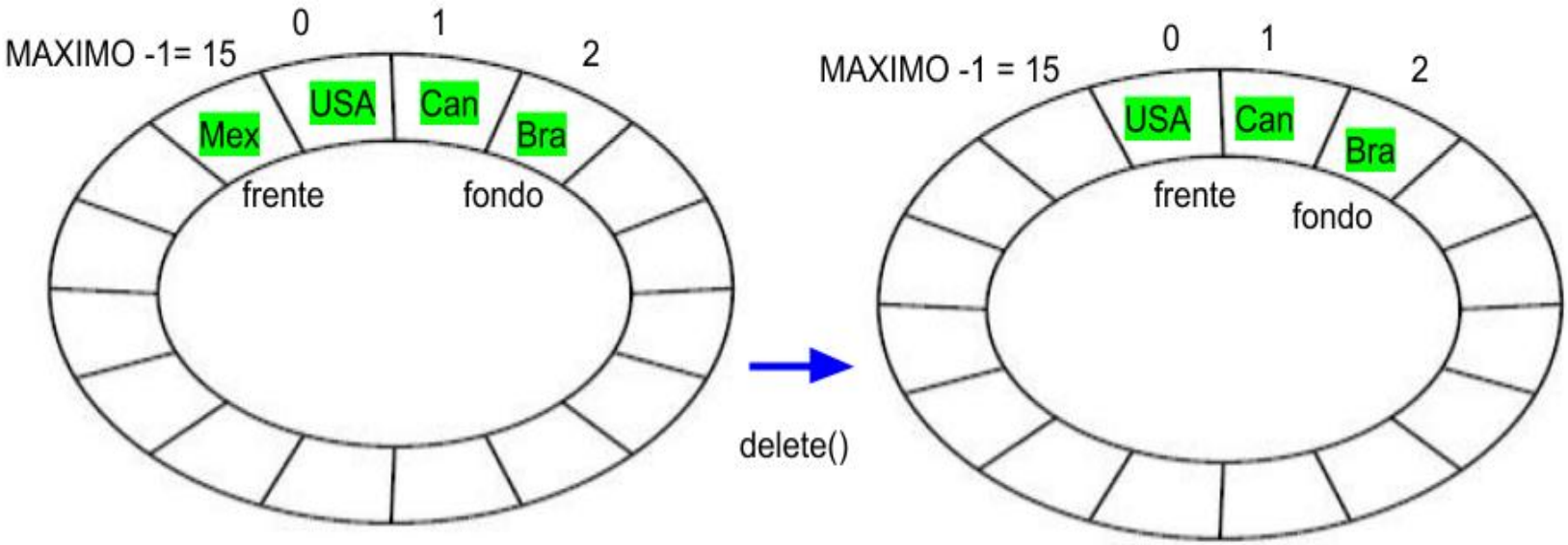
Cuando el fondo tiene el valor MAXIMO -1, una siguiente inserción provoca que el fondo se mueva a la posición 0, lo que refleja la característica circular del arreglo.



fondo = (fondo + 1) % MAXIMO = 16 % 16 = 0

Implementación de una cola basada en un arreglo circular

Cuando el frente tiene el valor MAXIMO -1, una siguiente eliminación provoca que el frente se mueva a la posición 0, lo que refleja de igual forma la característica circular del arreglo.



frente = (frente + 1) % MAXIMO = 16 % 16 = 0

Implementación de una cola basada en un arreglo circular

Constructor

Se establece que el frente sea inicialmente la primera posición y el fondo puede ser la última del arreglo. Como la siguiente operación que puede modificar los índices es una inserción forzosamente, esto moverá el fondo a la posición cero y apuntará al mismo lugar que el frente: el primer y único elemento de la cola.

```
/**
 * Construye una cola sin elementos.
 */
public ArregloCircular() {
    cola = new Object[MAXIMO];
    frente = 0;
    fondo = MAXIMO - 1;
    numeroElementos = 0;
}
```

Implementación de una cola basada en un arreglo circular

isEmpty() y size() quedan de forma natural en una línea. Para obtener el frente basta con acceder a la localidad referida por el índice respectivo.

```
@Override
public boolean isEmpty() {
    return numeroElementos == 0;
}

@Override
public int size() {
    return numeroElementos;
}

@Override
public Object front() {
    return isEmpty() ? null : cola[frente];
}
```

Implementación de la inserción

Se lanza excepción cuando la cola está llena y en caso contrario, se incrementa de forma circular (modular) el fondo en una unidad, se almacena el objeto nuevo y se incrementa el tamaño de la cola.

```
@Override
public void insert(Object obj) {
    if (numeroElementos == MAXIMO) {
        throw new RuntimeException("Cola llena");
    }
    fondo = (fondo + 1) % MAXIMO;
    cola[fondo] = obj;
    numeroElementos++;
}
```

Implementación de la eliminación

Cuando la cola no es vacía, se rescata el elemento que está al frente para poderlo regresar y antes de ésto último, se nulifica la localidad en el frente (por limpieza) se incrementa de forma circular (modular) el frente en una unidad y se decrementa el tamaño de la cola.

```
    @Override
    public Object delete() {
        if (isEmpty()) {
            return null;
        }
        Object tmp = cola[frente];
        cola[frente] = null;
        frente = (frente + 1) % MAXIMO;
        numeroElementos--;
        return tmp;
    }
}
```

Análisis de complejidad

Considerando como tamaño de la cola n el tamaño del arreglo circula que se emplea como atributo:

- Las operaciones size() y isEmpty() tienen complejidad constante porque el número de operaciones no dependen de n
- La operación front requiere en el peor caso un acceso al frente, lo cual es $O(1)$.
- Las operaciones insert(obj) y delete() acceden respectivamente al fondo y al frente, además de otro número constante de operaciones elementales, por lo que la complejidad en ambas es $O(1)$.

La desventaja es el desperdicio de espacio y que el tamaño del arreglo es fijo, pero la eficiencia por el acceso directo es mejor que si se emplea una lista ligada. Con una lista doblemente ligada como estructura de datos también es posible realizar todas las operaciones del TDA Cola con complejidad constante y con menos memoria que un arreglo circular.