Discussão da Implementação de Stacks com arrays

- A complexidade das operações é

função	Complexidade
IsEmpty	$O(1)$
Top	$O(1)$
Pop	$O(1)$
Push	$O(1)$

- A complexidade espacial é $O(N)$, independentemente do número de elementos que a stack contenha

- A desvantagem desta implementação é a fixação dum limite para o número de elementos que a stack comporta

  - **Solução??**

Queues

O TAD queue: sua definição e implementação com arrays circulares

25-Março-2021

O TAD QUEUE

- Trata-se doutra estrutura de dados fundamental (é da família das Stack)

- Os dados são acedidos usando o protocolo **FIFO** (First In First Out).

  - Nas Stacks o protocolo era **LIFO**(Last In First Out)

- É usual dizer-se que numa fila os elementos são inseridos no fim e removidos do início.

  - Trata-se duma metáfora derivada do vocabulário usado nas filas (da cantina, do supermercado)

O TAD QUEUE

- Em computação:
  - Este Tad define uma sequência de objectos estando restrita a inserção no fim/cauda da sequência e a remoção no início/frente
- FIFO

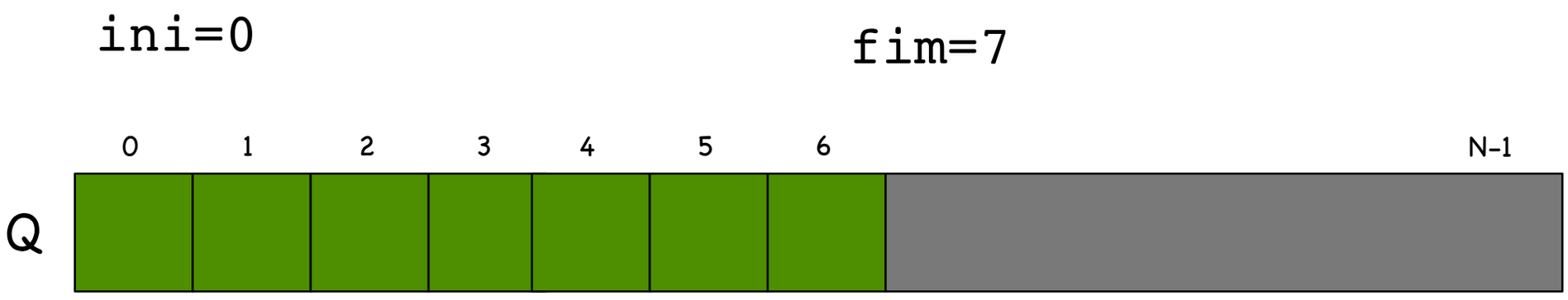
Especificação do TAD

```
typedef int ElementType;

        #ifndef _Queue_h
        #define _Queue_h

        struct QueueRecord;
        typedef struct QueueRecord *Queue;

        int IsEmpty( Queue Q );
        int IsFull( Queue Q );
        Queue CreateQueue( int MaxElements );
        void DisposeQueue( Queue Q );
        void MakeEmpty( Queue Q );
        void Enqueue( ElementType X, Queue Q );
        ElementType Front( Queue Q );
        ElementType Dequeue( Queue Q );

      #endif  /* _Queue_h */
```

Implementação com um array

- Tal como foi feito com as Stacks as Queues podem ser implementadas com arrays, prefixando o tamanho máximo da fila.

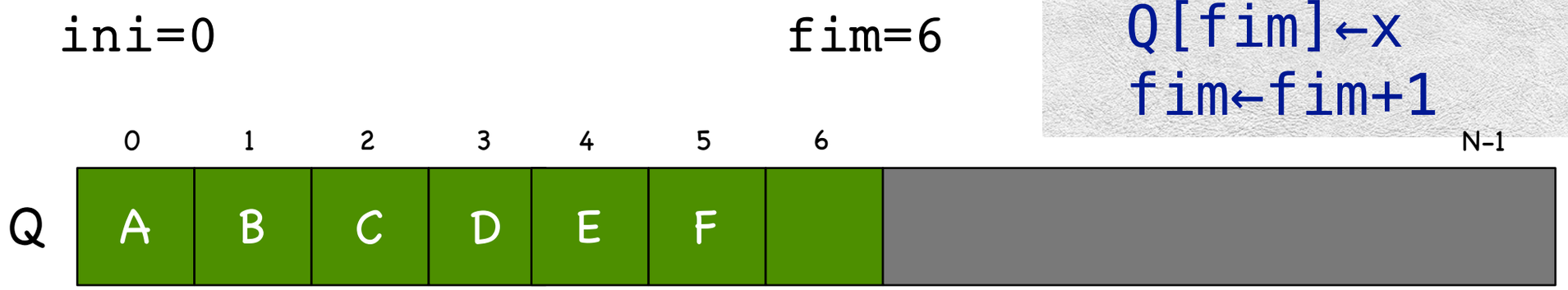
- Agora há que decidir como gerir o início e o fim da fila. Uma hipótese é considerar como início a primeira posição do array e o fim a última inserção



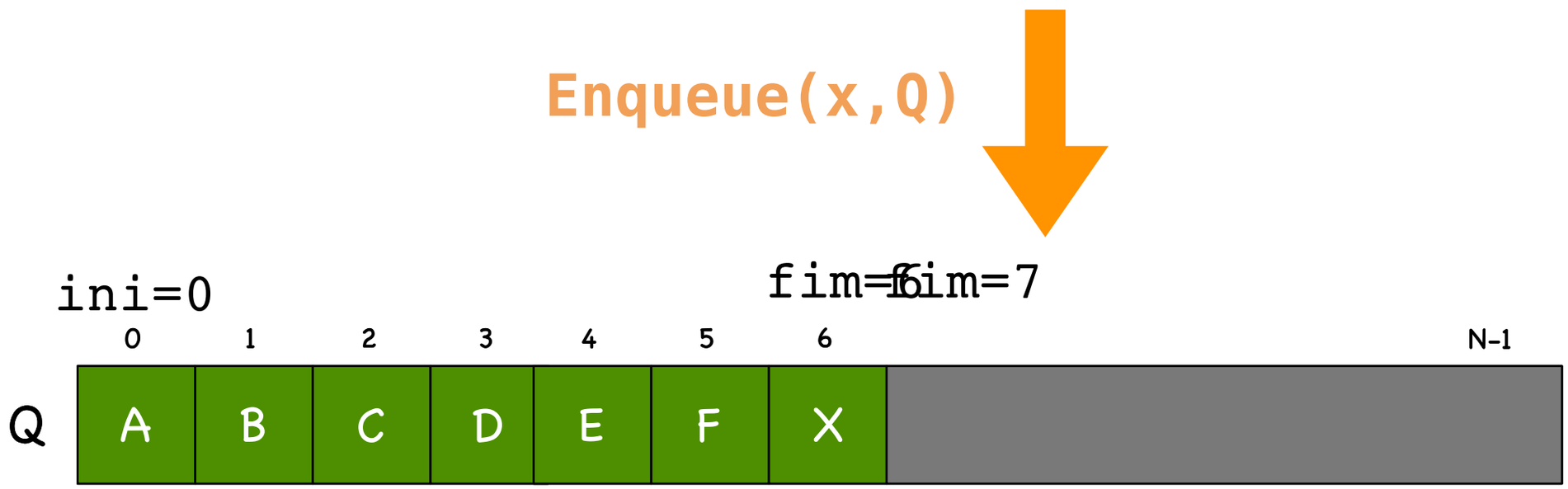
Implementação com um array

- Como gerir a inserção?

```
Algoritmo Enqueue(x,Q):
   if Size(Q)=N
      Error FullException

   Q[fim]←x
   fim←fim+1
```

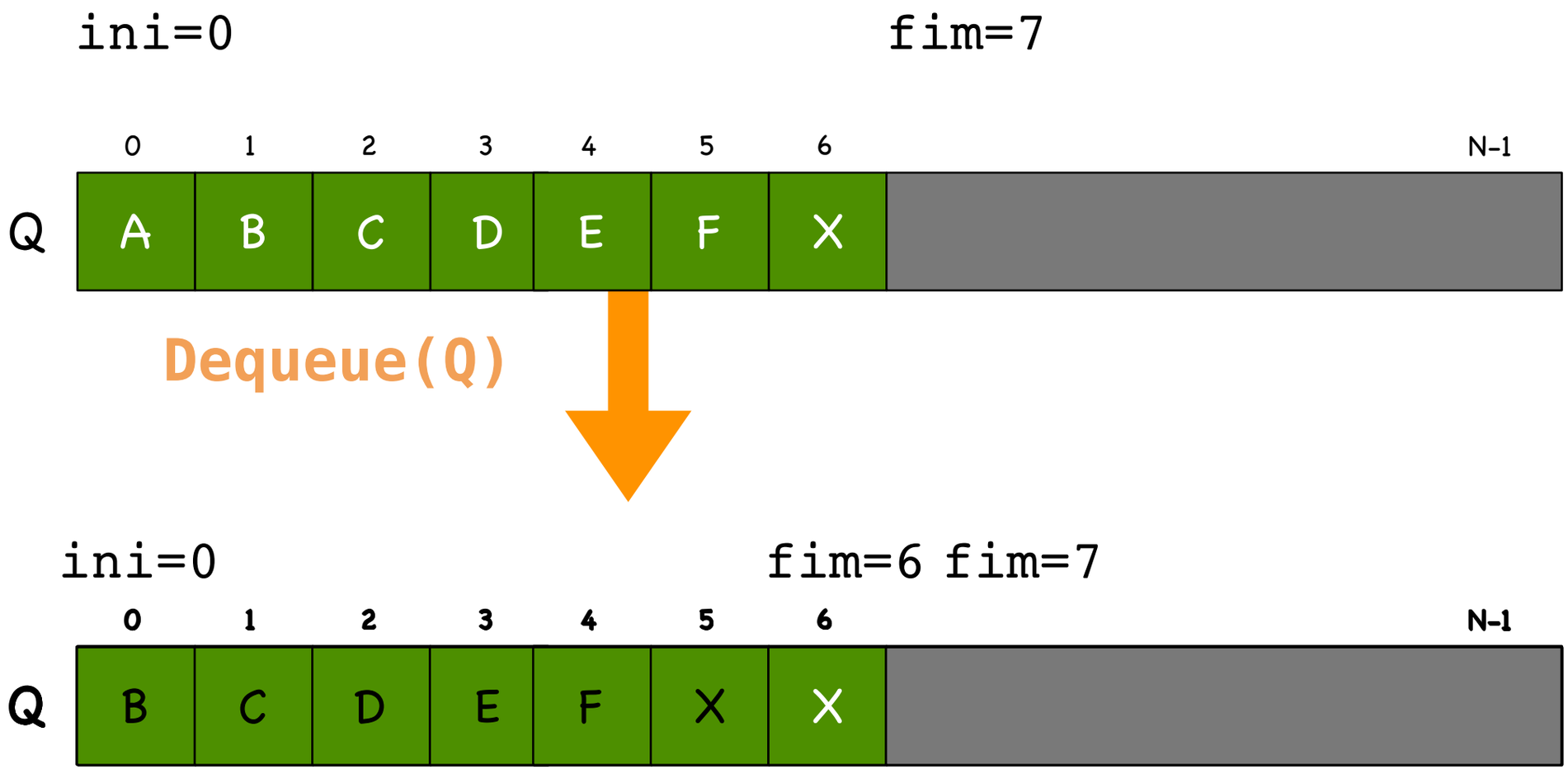


Enqueue(x,Q)



Implementação com array

- Como gerir a remoção?



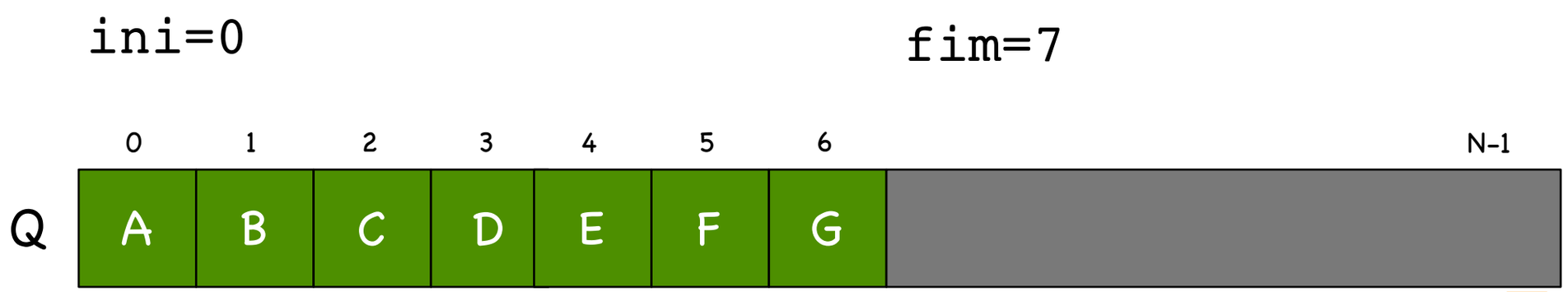
Implemetação com um array

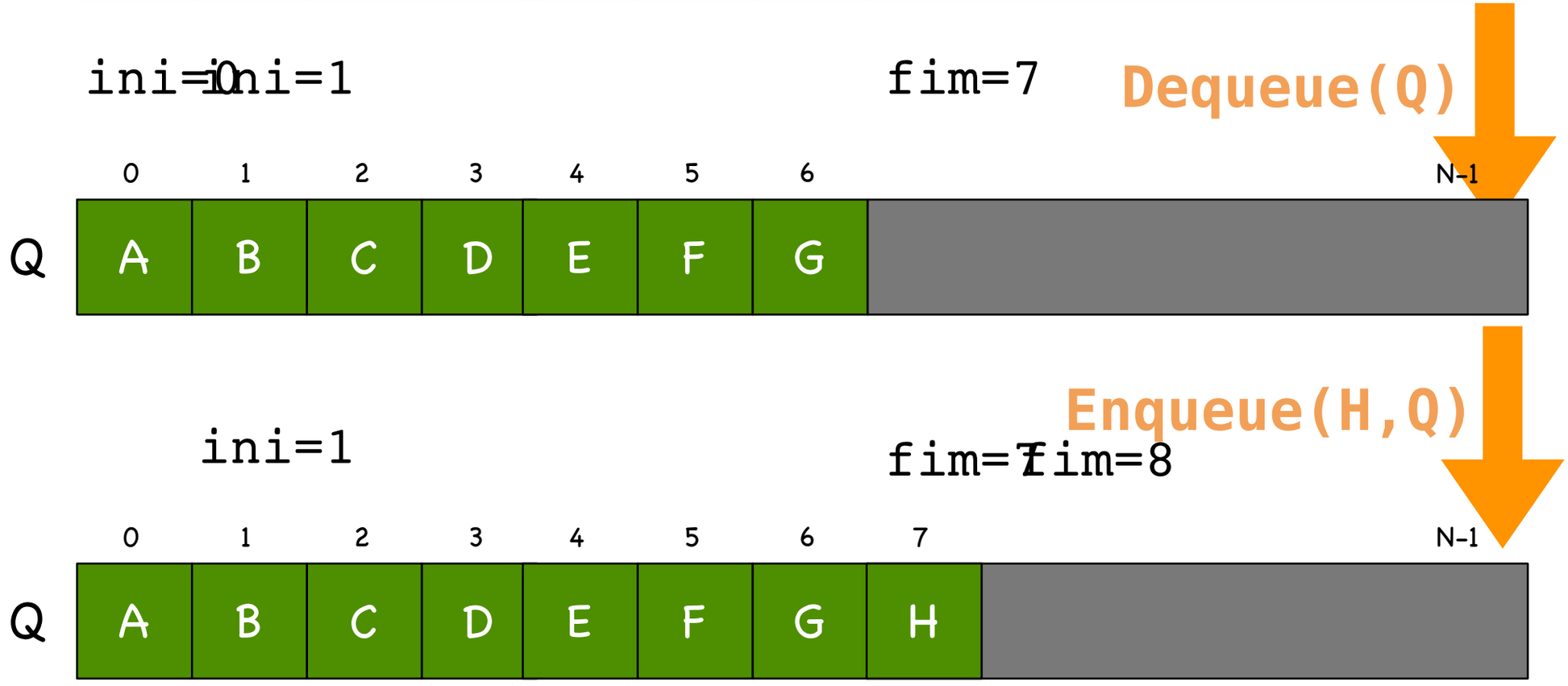
- Temos de deslocar uma posição todos os n elementos que estiverem na fila, sempre! Trata-se duma operação de complexidade $\theta(n)$

- Para obter um tempo constante para a operação Dequeue temos que ser mais elásticos

Implementação com array

- O primeiro elemento da fila não necessita de ser o índice zero(0)





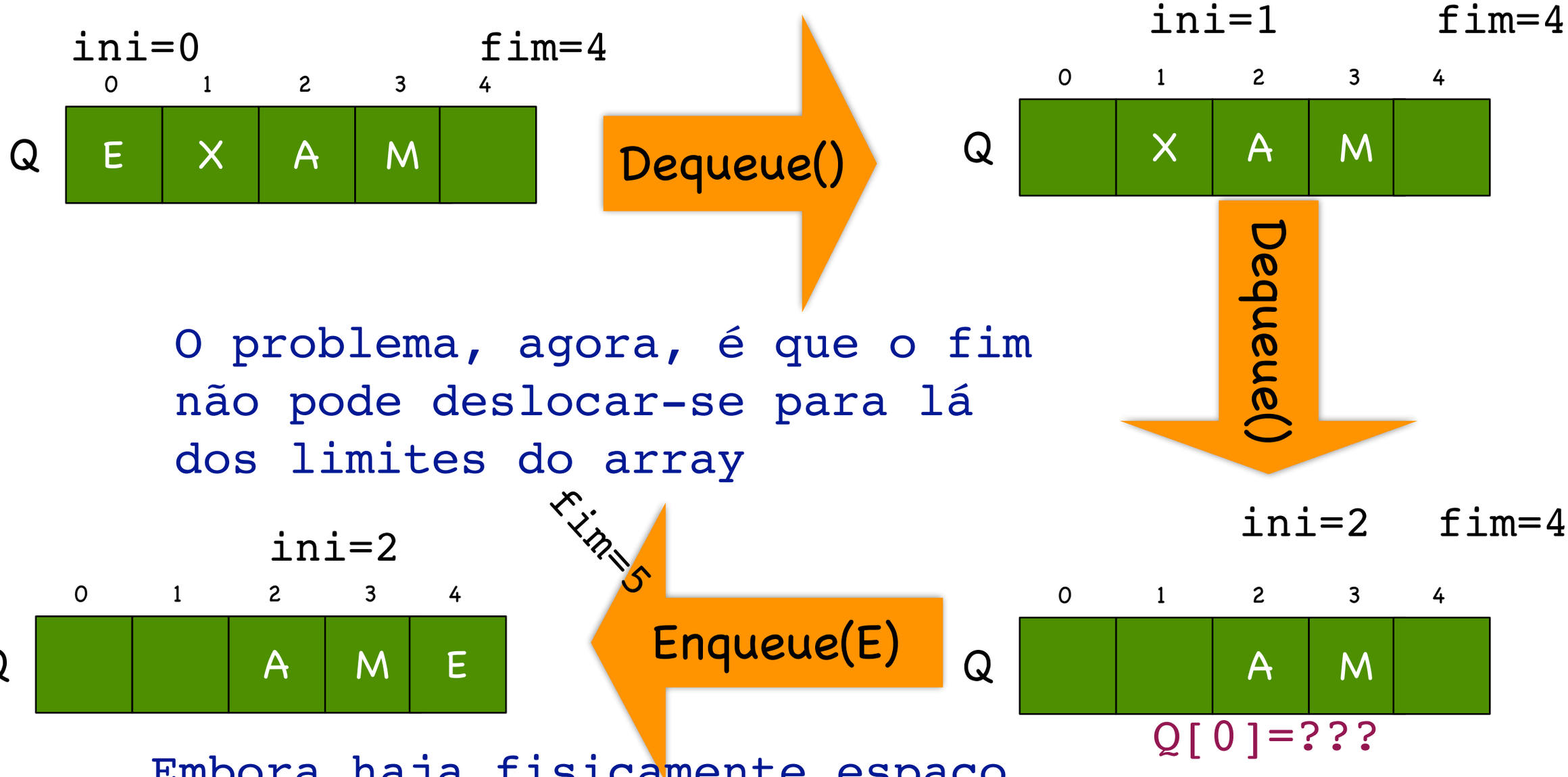
Implementação com um array

- Quando se adiciona um elemento à fila (enqueue) o fim da fila desloca-se para a frente em direcção ao fim do array

- Quando se remove um elemento da fila (dequeue) o inicio da fila desloca-se para a frente em direcção ao fim da fila, não havendo necessidade de realizar cópias dos elementos

Implementação com um array

- Seja Q uma queue de tamanho 5, faça-se: Enqueue(E); Enqueue(X); Enqueue(A); Enqueue(M)



O problema, agora, é que o fim não pode deslocar-se para lá dos limites do array

Embora haja fisicamente espaço para inserir…

Implementação com um array circular

- Esta **técnica** designa-se por implementação com um array circular.

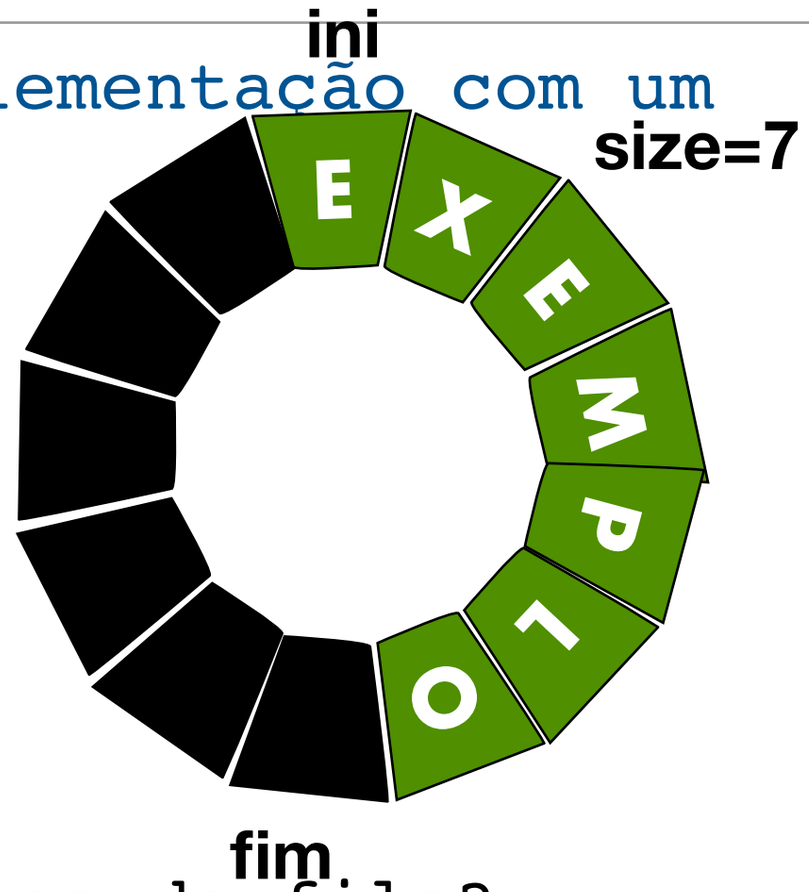
- Problemas?

  - Como saber se a fila cheia?

  - E vazia?

  - Como saber o número de elementos da fila?

  - Como incrementar as variáveis ini e fim



ini=2

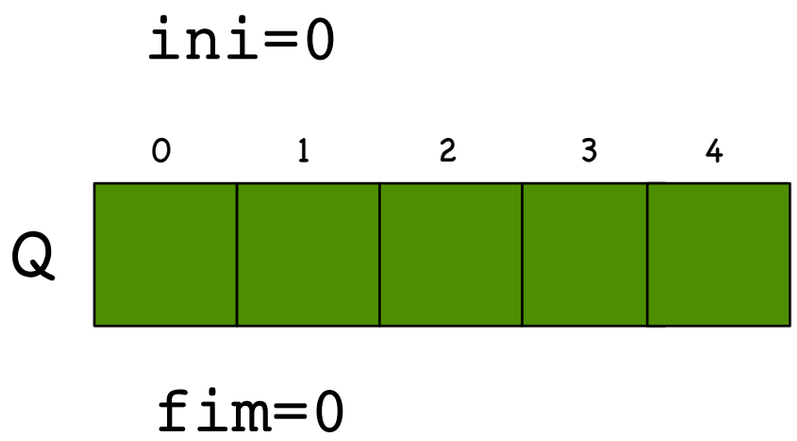
fim=1

Q

0	1	2	3	4
I		A	M	E

Vazia ou cheia

- Como saber se a fila está vazia?
  - ini=fim? (ini=0, fim=0 no início!)
- Como saber se a fila está cheia?
  - ini=fim?
  - igual???



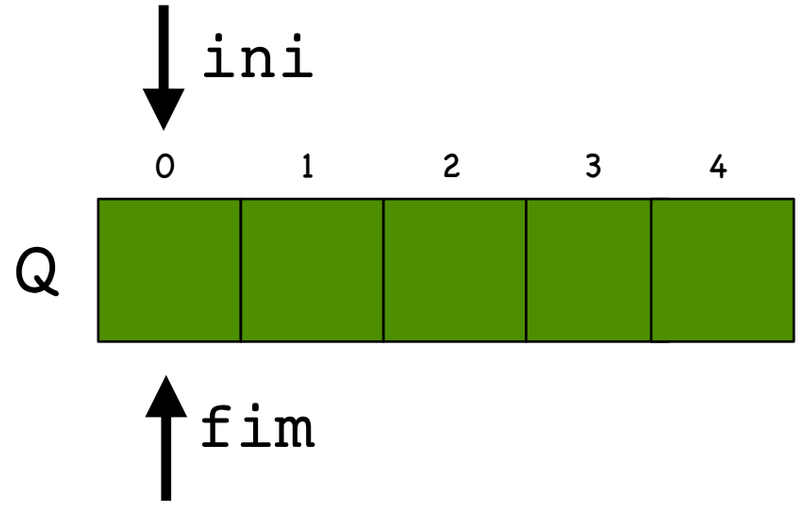
Vazia ou cheia

- Soluções:
  - Usar uma variável booleana para saber se está ou não vazia
  - Arranjar uma array com N+1 posições e só permitir que N posições sejam ocupadas
  - Arranjar um contador do número de elementos da Queue

Implementação com array

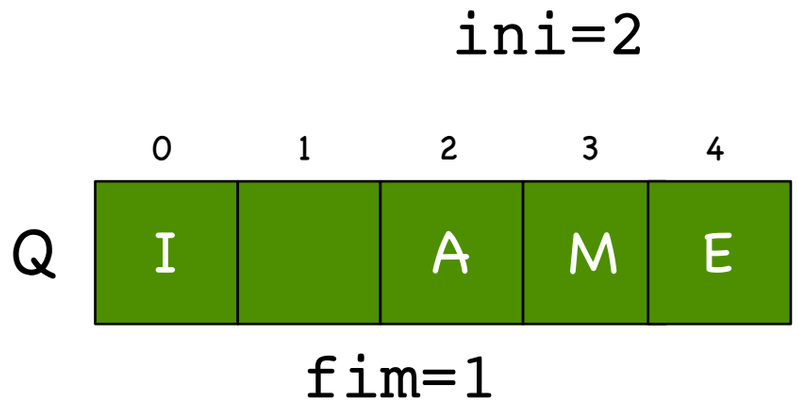
- Usando um array de dimensão N e só permitindo a inserção de N-1 elementos:
  - Um teste para saber se a queue está vazia é saber se ini é coincidente com fim.
  - Para saber se está cheia é saber o número de elementos é N-1. O numero de elementos pode ser calculado :

size=(N-ini+fim) mod N



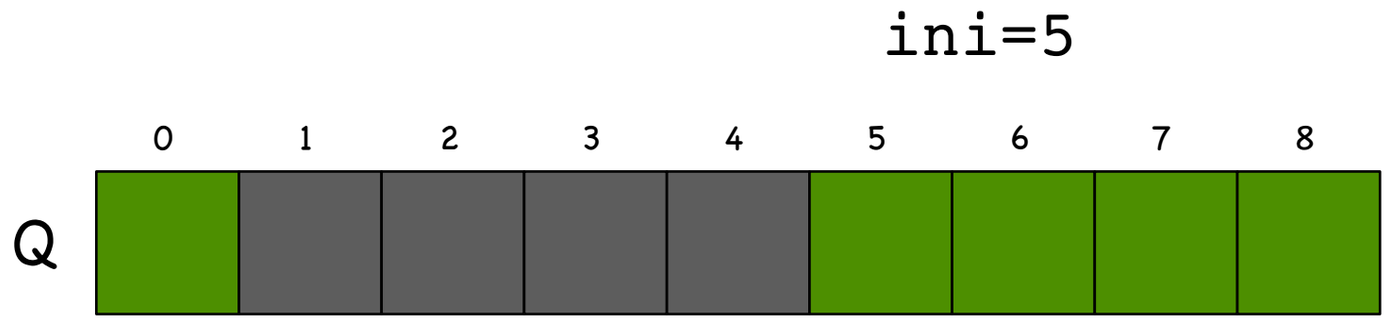
size=(5-0)%5=0

Implementação com array

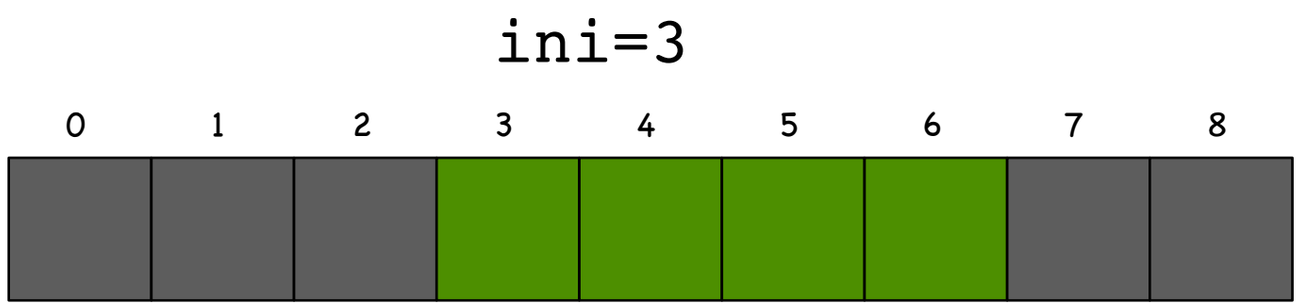


size=(5-2+1) % 5=4

Cheia!!



size=(9-5+1) % 9 = 5



size=(9-3+7) % 9 = 4

```
Algoritmo size():
   return (N-ini+fim) mod N
```

```
Algoritmo inc(i):
   return (i+1) mod N
```

```
Algoritmo empty():
   return ini=fim
```

complexidade das operações?

```
Algoritmo enqueue(e):
   if size()=N-1
      Error Overflow

   Q[fim]←e
   fim←inc(fim)
```

```
Algoritmo dequeue():
   if empty()
      Error Empty Queue
   x←Q[ini]
   inc←inc(ini)
   return x
```

Exercício

- Escreva uma função que receba como argumento uma Queue e retorne uma Queue com os mesmos elementos mas por ordem inversa

```
Queue inverte(Queue X){
    Stack S1=CreateStack(10);
    while (!IsEmptyQ(X)){
        Push(Dequeue(X),S1);
    }
    MakeEmptyQ(X);
    while (!IsEmpty(S1)){
        Enqueue(Pop(S1),X );
    }

    return X;

}
```