

O TAD fila

- ◆ O TAD **fila** armazena objetos arbitrários
- ◆ Inserções e remoções seguem o esquema FIFO
- ◆ Inserções são feitas no fim da fila e remoções no início da fila
- ◆ Operações principais:
 - **enqueue**(object): insere um elemento no fim da fila
 - object **dequeue**(): remove e retorna o elemento do início da fila

- ◆ Operações auxiliares:
 - object **first**(): retorna o elemento do início sem removê-lo
 - integer **size**(): retorna o número de elementos armazenados
 - boolean **isEmpty**(): indica se há elementos na fila

◆ Exceções

- Tentar remover o ver um elemento do início da fila levanta a exceção **EFilaVazia**

Aplicações de fila

◆ Aplicações diretas

- Filas de esperas
- Acesso a recursos compartilhados (impres.)
- Programação paralela

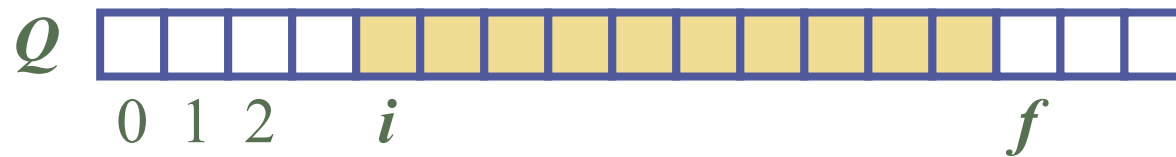
◆ Aplicações indiretas

- Estrutura de dados auxiliar para algoritmos
- Componentes de outras estruturas de dados

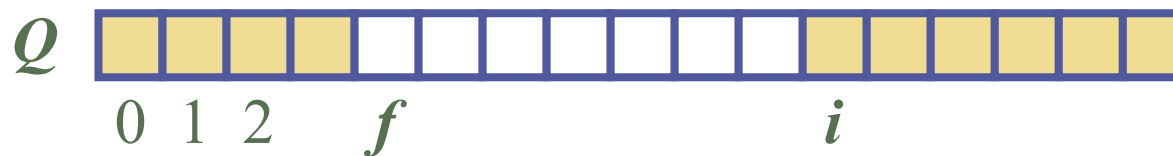
Fila baseada em *array*

- ◆ Use um *array* de tamanho N de forma circular
- ◆ Duas variáveis mantêm informações do início e fim da fila
 - i índice do elemento do início
 - f índice imediatamente após o último elemento
- ◆ A posição f no array fica vazia

Configuração normal



Configuração "quebrada"

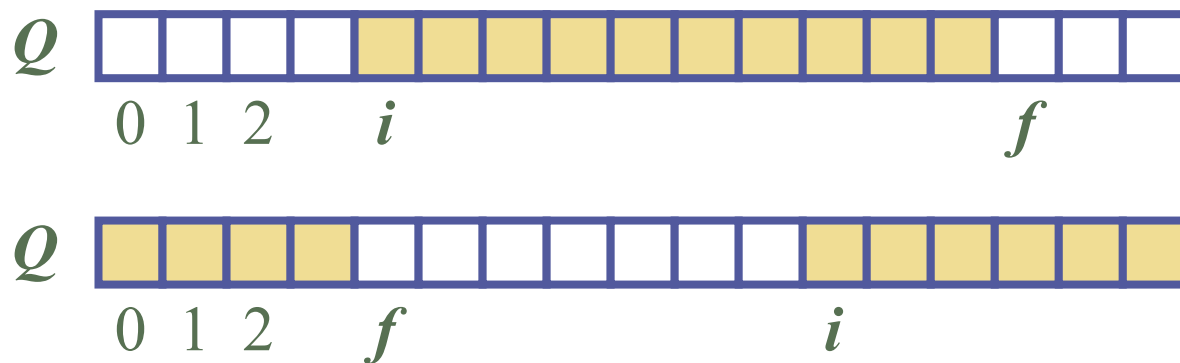


Operações sobre o TAD fila

◆ Usamos o operador módulo (resto da divisão)

Algoritmo *tamanho()*
retorne $(N - i + f) \% N$

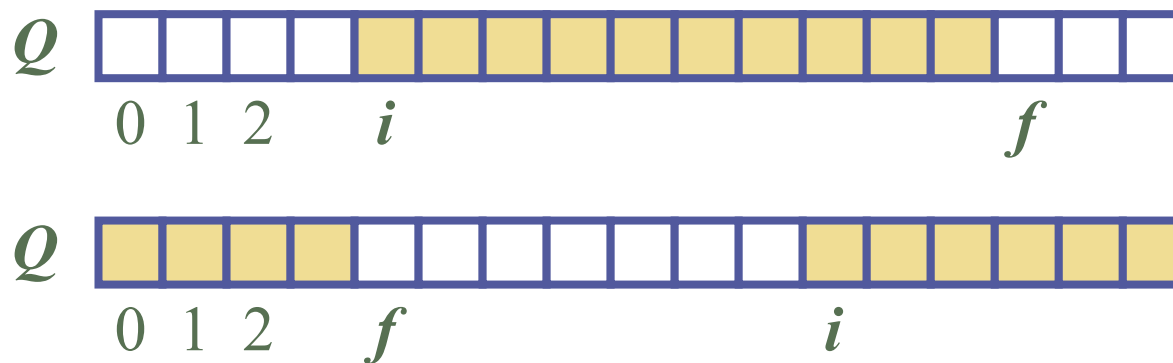
Algoritmo *estaVazia()*
return $(i = f)$



Operações sobre o TAD fila

- ◆ Operação *enqueue* levanta uma exceção se o *array* está cheio
- ◆ Esta exceção é dependente da implementação

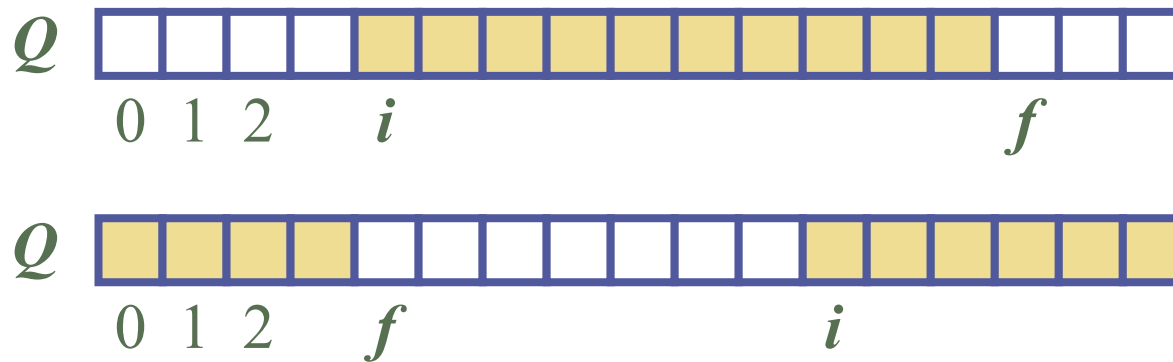
```
Algoritmo enqueue(o)  
  Se (size() =  $N - 1$ ) então  
    throw EFilaCheia  
  senão  
     $Q[f] \leftarrow o$   
     $f \leftarrow (f + 1) \% N$ 
```



Operações sobre o TAD fila

- ◆ Operação *desenfileirar* levanta uma exceção se a fila está vazia
- ◆ Esta exceção é específica do TAD Fila

```
Algoritmo dequeuear()  
Se (isEmpty()) então  
    throw EFilaVazia  
senão  
     $o \leftarrow Q[i]$   
     $i \leftarrow (i + 1) \% N$   
    retorne  $o$ 
```



Fila crescente baseada em *array*

- ◆ Em uma operação enqueue(*enfileirar*), quando o *array* está cheio, ao invés de levantar uma exceção, podemos substituir o *array* por um maior
- ◆ Similar ao que fizemos com Pilhas
- ◆ A operação *enfileirar* tem tempo de execução amortizado
 - $O(n)$ com estratégia incremental
 - $O(1)$ com estratégia de duplicação

Interface fila em JAVA

- ◆ Interface em JAVA que corresponde ao nosso TAD
- ◆ Requer a definição da classe `EFilaVazia`

```
public interface Fila {  
    public int tamanho();  
    public boolean estaVazia();  
    public Object inicio()  
        throws EFilaVazia;  
    public void enfileirar(Object o);  
    public Object desenfileirar()  
        throws EFilaVazia;  
}
```

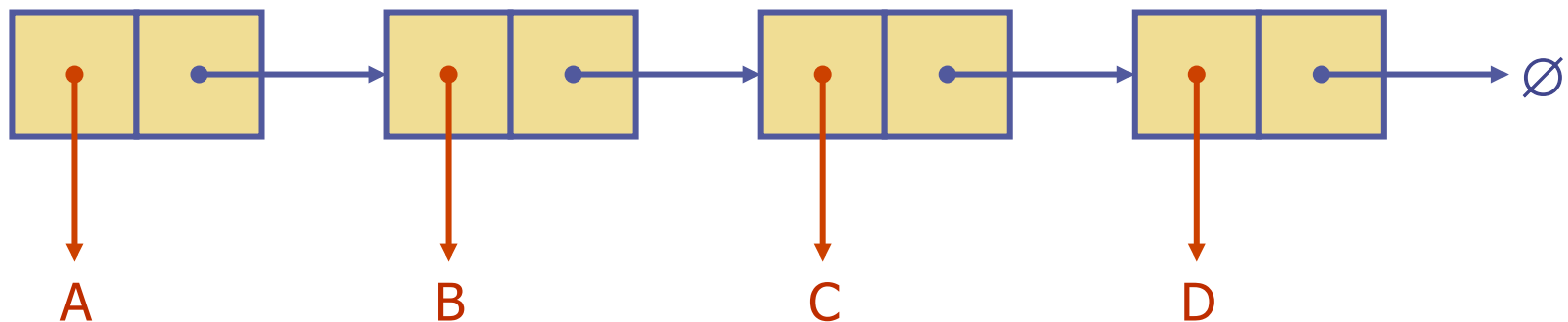
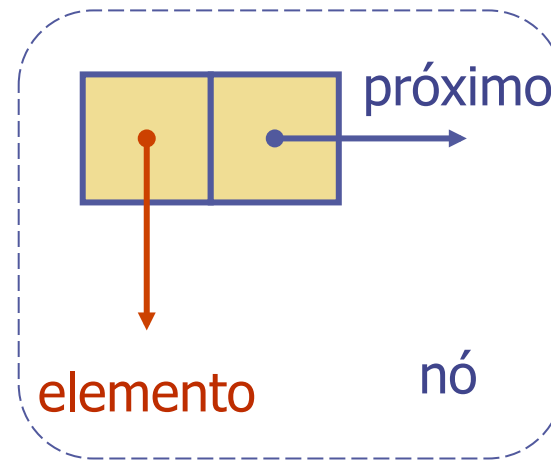

O TAD deque

- ◆ O TAD deque armazena objetos arbitrários
- ◆ Inserções e remoções podem ser feitas no início ou no fim
 - É uma fila de duplo sentido

- ◆ Operações principais:
 - `inserirInicio(object):`
 - `object removerInicio():`
 - `inserirFim(object):`
 - `object removerFim():`
- ◆ Operações auxiliares
 - `object primeiro():`
 - `object ultimo():`
 - `int tamanho():`
 - `boolean estaVazia():`

Lista encadeada

- ◆ Uma lista encadeada é uma estrutura de dados concreta consistindo de uma sequência de nós
- ◆ Cada nó armazena
 - Um elemento
 - Uma *ligação* com o próximo nó

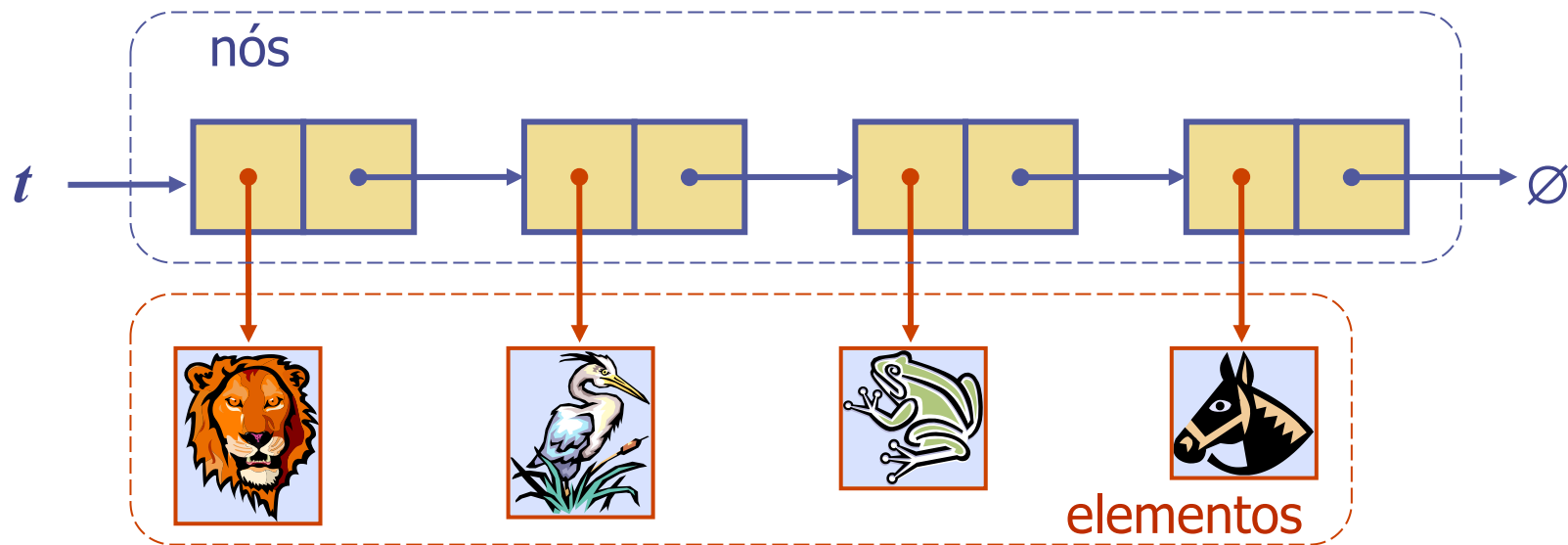


Classe No

```
public class No {  
    private Object elemento;  
    private No proximo;  
    public Object getElemento() {  
        return elemento;  
    }  
    public void setElemento(Object o){  
        elemento = o;  
    }  
}
```

Pilhas com listas encadeadas

- ◆ Pode-se implementar uma pilha com uma lista encadeada
- ◆ O elemento do topo é armazenado no primeiro nó
- ◆ O espaço usado é $O(n)$ e cada operação roda em tempo $O(1)$



Filas com listas ligadas

- ◆ Pode-se implementar uma fila com uma lista ligada
 - O elemento do início é o primeiro nó
 - O elemento do fim é o último nó
- ◆ O espaço usado é $O(n)$ e cada operação roda em tempo $O(1)$

