

# ED62A-COM2A

# ESTRUTURAS DE DADOS

Aula 04A - Filas  
Implementação dinâmica

Prof. Rafael G. Mantovani  
09/04/2019

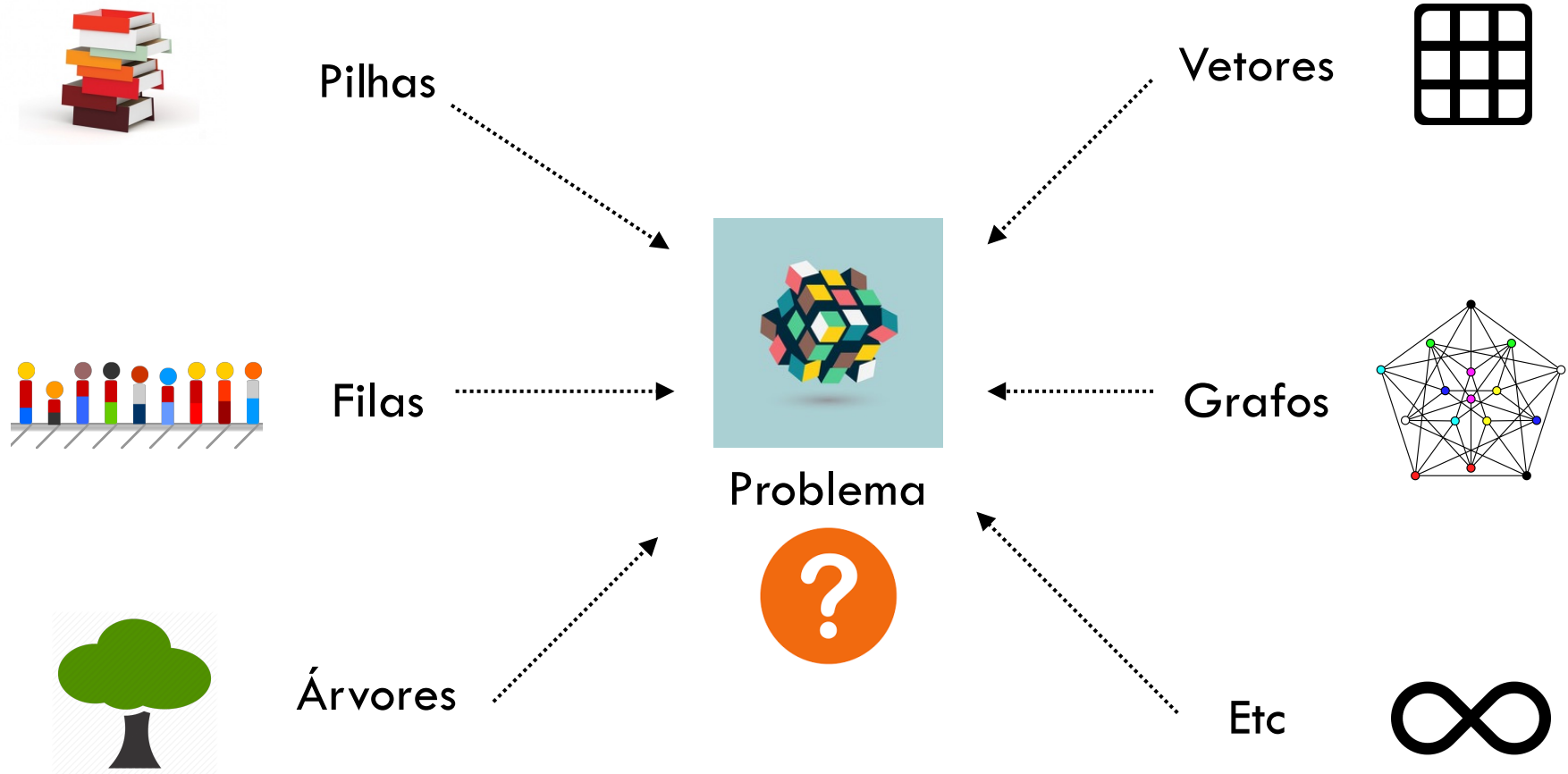
# Roteiro

- 1** Introdução
- 2** Filas
- 3** Operações gerais
- 4** Inserção de elementos
- 5** Remoção de elementos
- 6** Referências

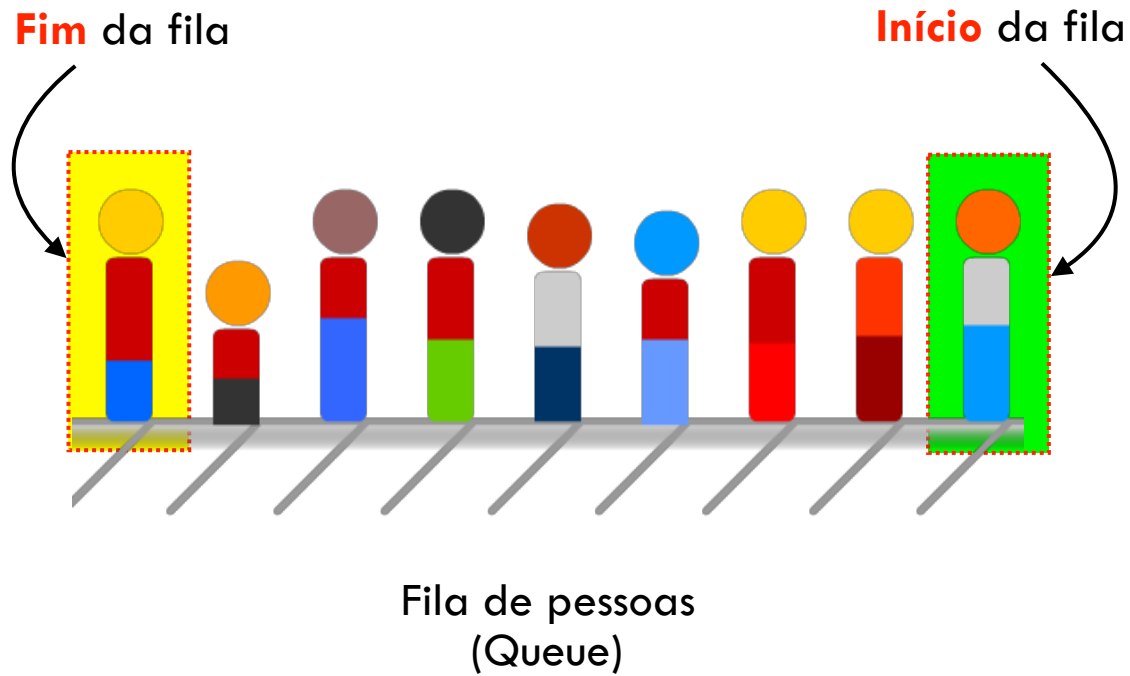
# Roteiro

- 1** Introdução
- 2** Filas
- 3** Operações gerais
- 4** Inserção de elementos
- 5** Remoção de elementos
- 6** Referências

# Introdução



# Filas

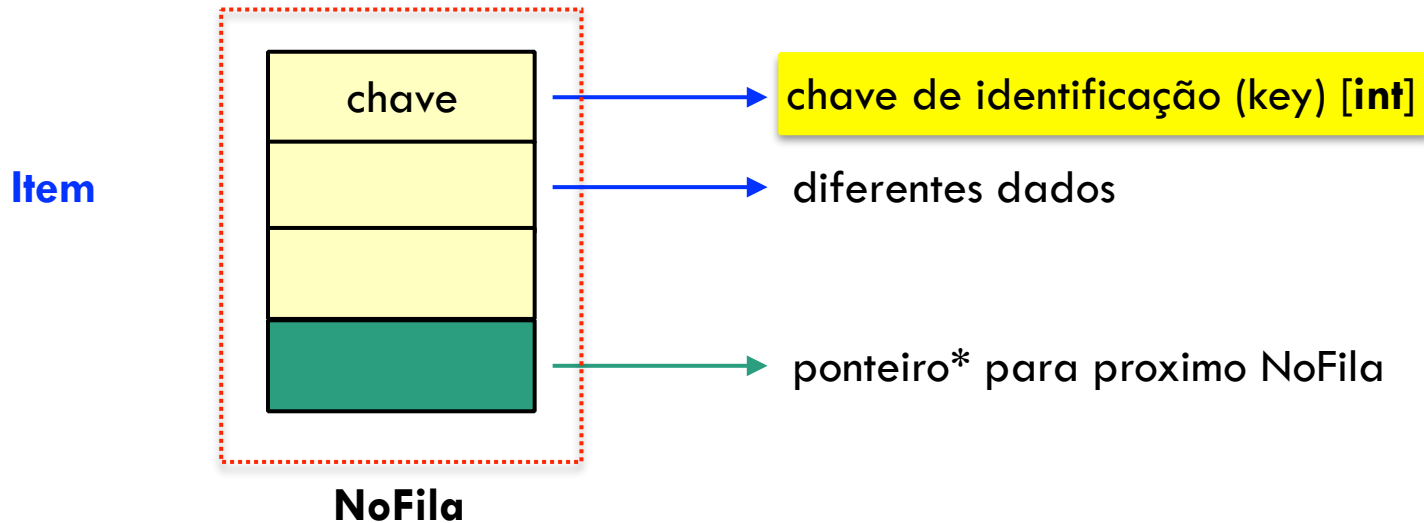


# Roteiro

- 1 Introdução
- 2 Filas dinâmica
- 3 Operações gerais
- 4 Inserção de elementos
- 5 Remoção de elementos
- 6 Referências

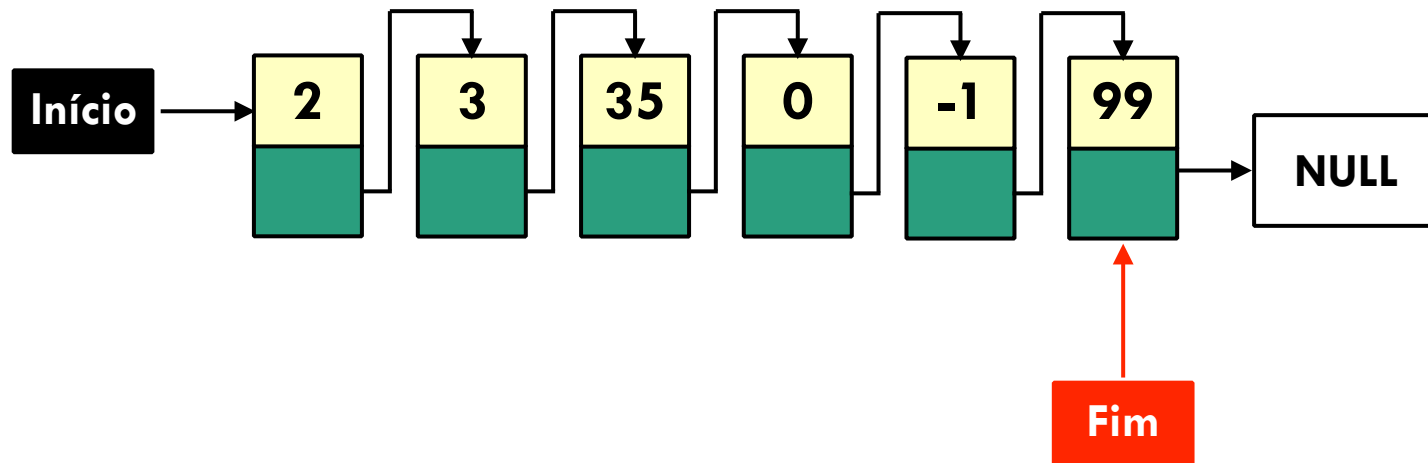
# Fila dinâmica

- Nós de Fila (estrutura dinâmica)



# Fila dinâmica

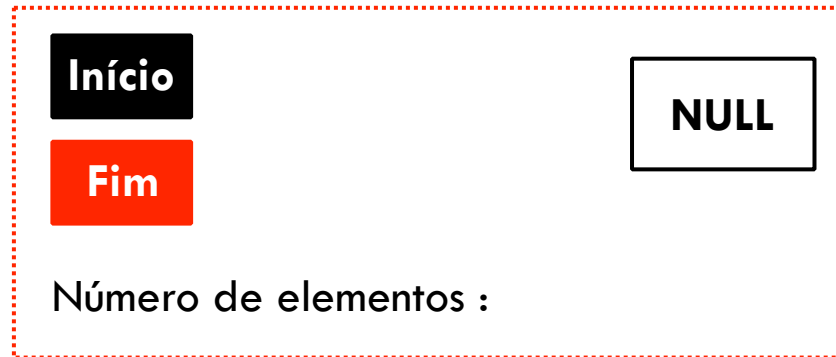
Número de elementos : 6





# Fila dinâmica

**tipo Fila Dinâmica**



# Roteiro

- 1 Introdução
- 2 Filas
- 3 Operações gerais
- 4 Inserção de elementos
- 5 Remoção de elementos
- 6 Referências

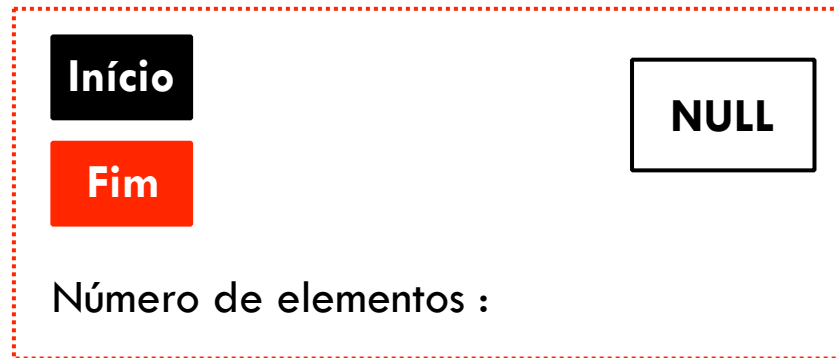
# Operações

Dada uma pilha  $S$ , chave  $k$ , elemento  $x$ :

- **iniciar/destruir** → **iniciar e destruir a fila**
- ~~pesquisar( $S, k$ ) → procurar  $k$  em  $S$  [TRUE/FALSE]~~
- **inserir( $S, k$ )** → **inserir  $k$  em  $S$**
- **remover( $S, k$ )** → **remover  $k$  em  $S$**
- ~~minimo( $S$ ) → menor valor armazenado em  $S$~~
- ~~maximo( $X$ ) → maior valor armazenado em  $S$~~
- ~~proximo( $S, x$ ) → elemento sucessor a  $x$~~
- ~~anterior( $S, x$ ) → elemento antecessor a  $x$~~
- **tamanho( $S$ )** → **tamanho de  $S$**
- **vazia( $S$ )** →  **$S$  está vazia? [TRUE/FALSE]**
- ~~cheia( $S$ ) →  $S$  está cheia? [TRUE/FALSE]~~
- **primeiro( $S$ )** → **elemento no início da fila**
- **último( $S$ )** → **elemento no final da fila**

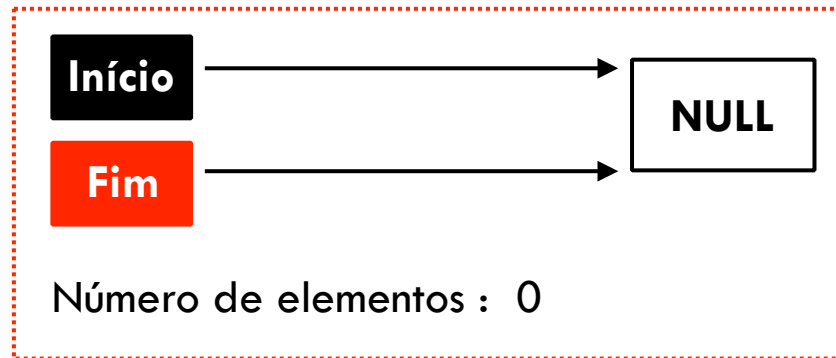
# Inicialização da fila

**tipo Fila Dinâmica**



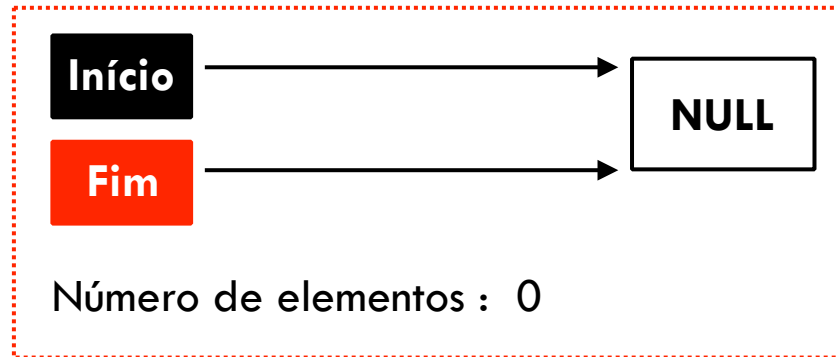
# Inicialização da fila

**tipo Fila Dinâmica**



# Inicialização da fila

## tipo Fila Dinâmica

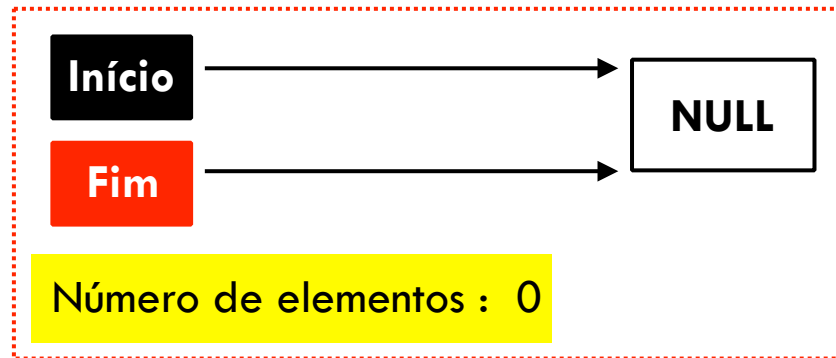


### IniciaFila (Q)

1. `Q.inicio = NULL;`
2. `Q.fim = NULL;`
3. `Q.tamanho = 0;`

# Inicialização da fila

**tipo Fila Dinâmica**

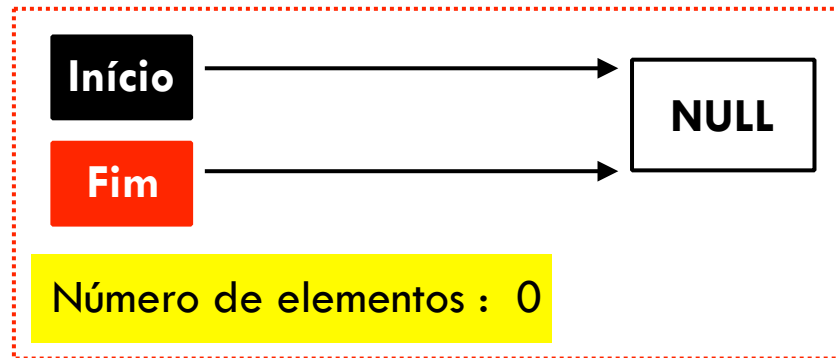


**tamanhoFila (Q)**

**estaVazia (Q)**

# Inicialização da fila

## tipo Fila Dinâmica



**tamanhoFila (Q)**  
1. return (Q.tamanho);

**estaVazia (Q)**  
1. return (Q.tamanho == 0);  
// return(Q.Inicio == NULL)



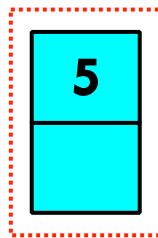
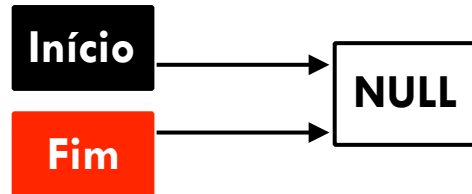
# Roteiro

- 1 Introdução
- 2 Filas
- 3 Operações gerais
- 4 Inserção de elementos
- 5 Remoção de elementos
- 6 Referências

# Inserção (Enqueue)

a) primeira inserção (elemento  $x = 5$ )

Número de elementos : 0

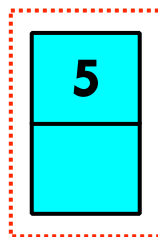
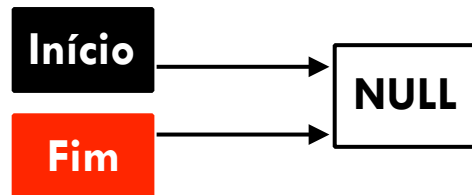


**NoFila**  
**(Aux)**

# Inserção (Enqueue)

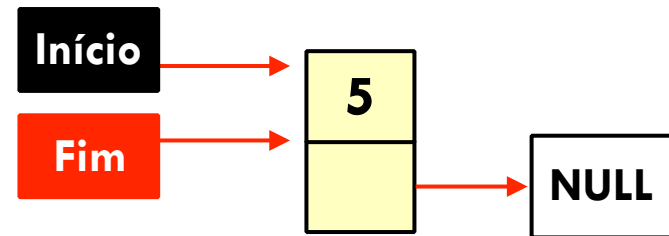
a) primeira inserção (elemento  $x = 5$ )

Número de elementos : 0



NoFila  
(Aux)

Número de elementos : 1

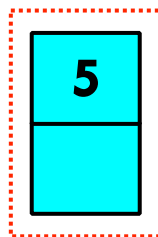
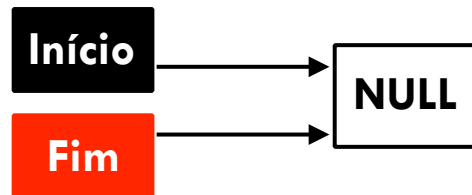


NoFila  
(Aux)

# Inserção (Enqueue)

a) primeira inserção (elemento  $x = 5$ )

Número de elementos : 0

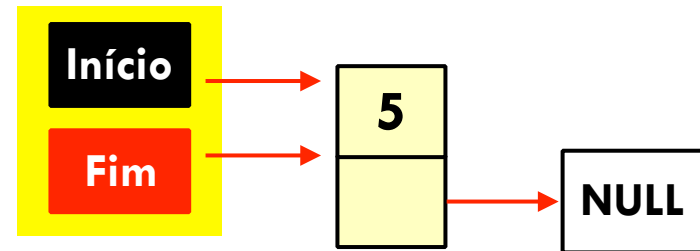


NoFila  
(Aux)

01

1. **Início** e **Fim** apontam para **Aux** (novo nó)
2. **Aux** aponta para NULL
3. contador é incrementado

Número de elementos : 1

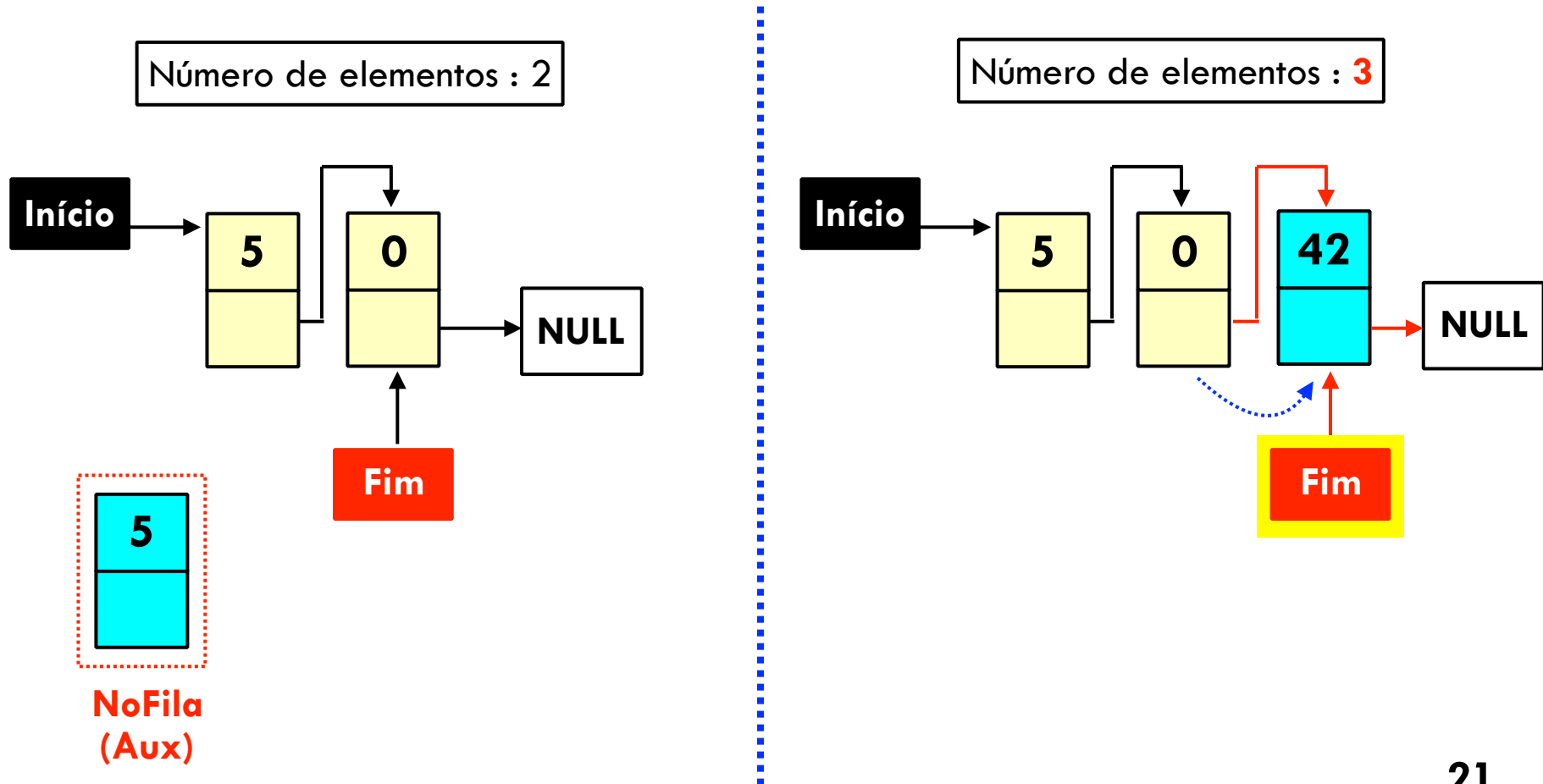


NoFila  
(Aux)

O que aconteceu?

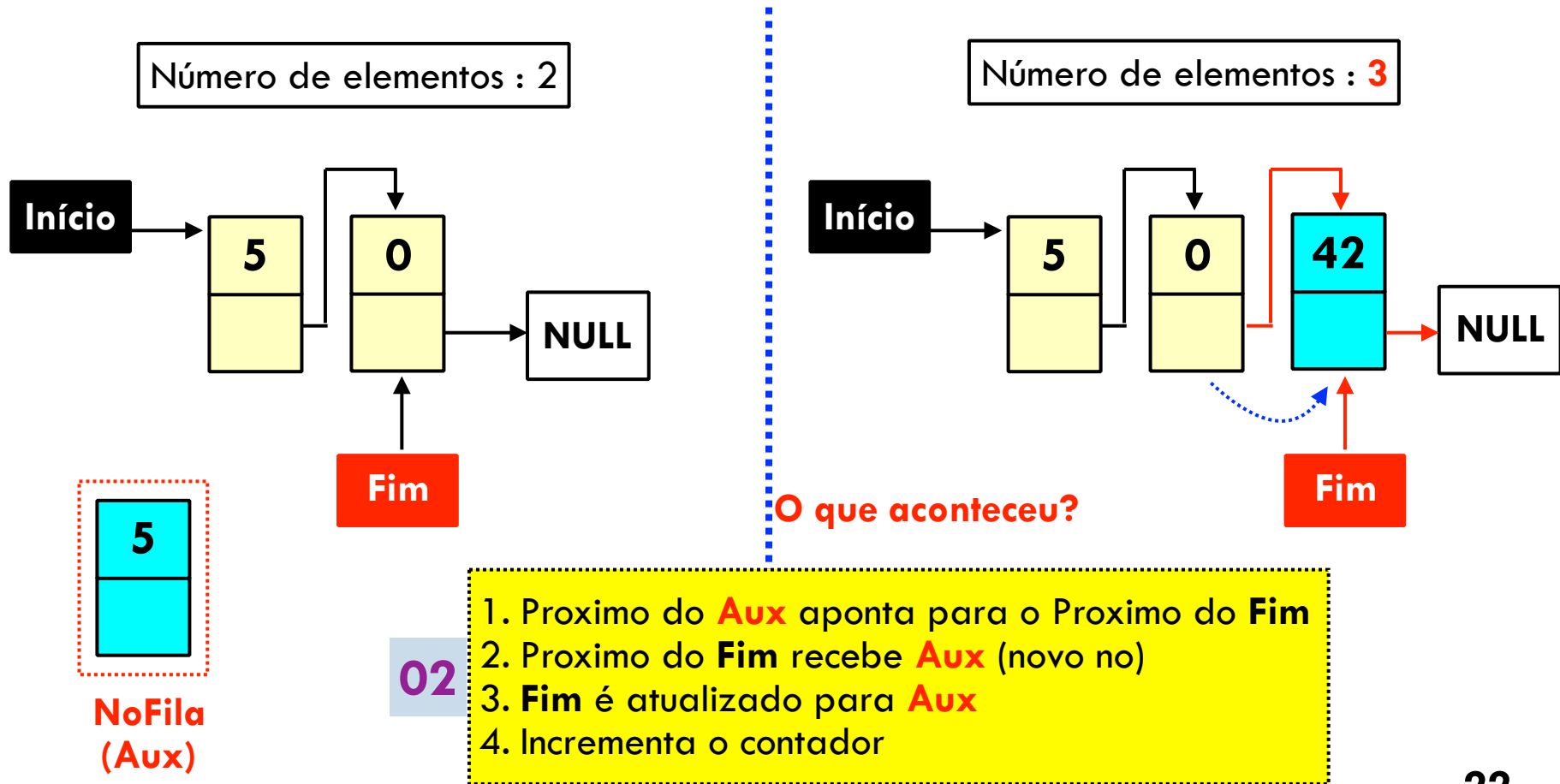
# Inserção (Enqueue)

b) não é primeira inserção (elemento  $x = 42$ )



# Inserção (Enqueue)

b) não é primeira inserção (elemento  $x = 42$ )



# Enfileirar (enqueue)

**Enqueue (Q, x)**

1. se for a primeira inserção:
2.    Q.Inicio = Q.Fim = Aux
3.    Aux->proximo = NULL
4. senão:
5.    Aux->proximo = NULL
6.    Fim->proximo = Aux
7.    Fim = Fim->Proximo // *Fim = Aux*
8. incrementa contador de elementos

# Enfileirar (enqueue)

Enqueue (Q, x)

1. se for a primeira inserção:

2.    Q.Inicio = Q.Fim = Aux

3.    Aux->proximo = NULL

4. senão:

5.    Aux->proximo = NULL

6.    Fim->proximo = Aux

7.    Fim = Fim->Proximo // *Fim = Aux*

8. incrementa contador de elementos

01

02



# Enfileirar (enqueue)

Enqueue (Q, x)

1. se for a primeira inserção:

2.    Q.Inicio = Q.Fim = Aux

3.    Aux->proximo = NULL

4. senão:

5.    Aux->proximo = NULL

6.    Fim->proximo = Aux

7.    Fim = Fim->Proximo // *Fim = Aux*

8. incrementa contador de elementos

Primeira  
inserção

01

02

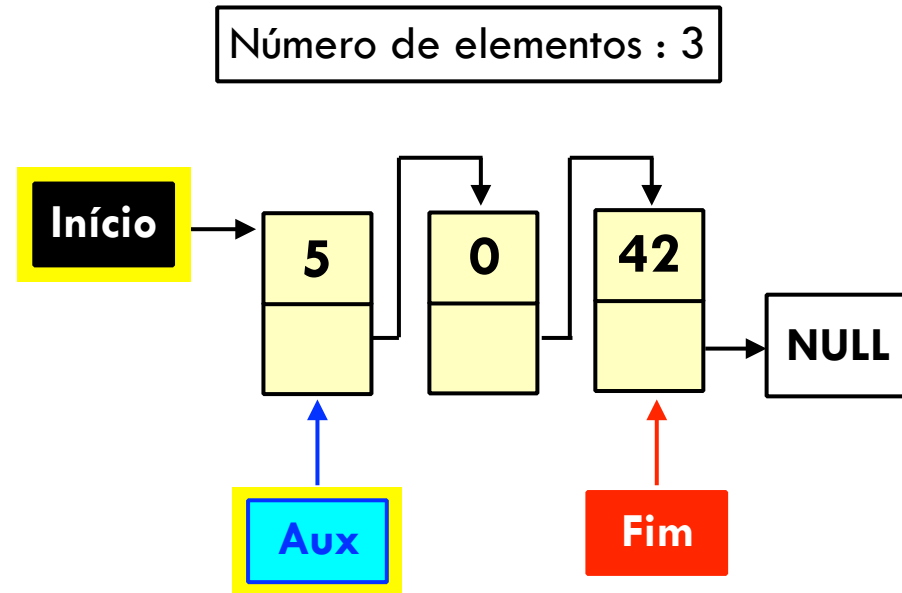
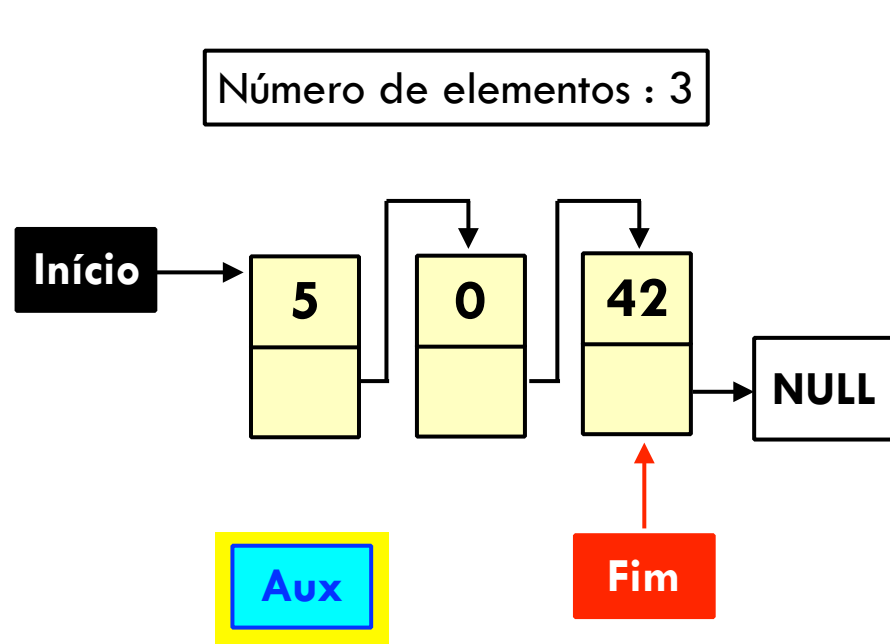
Já contém  
elementos

# Roteiro

- 1 Introdução
- 2 Filas
- 3 Operações gerais
- 4 Inserção de elementos
- 5 Remoção de elementos
- 6 Referências

# Remoção (dequeue)

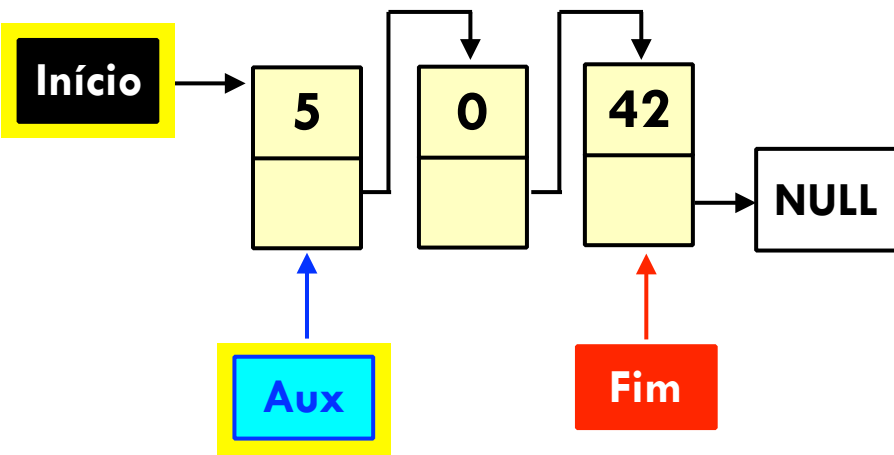
b) desenfileirar (remover elemento 42)



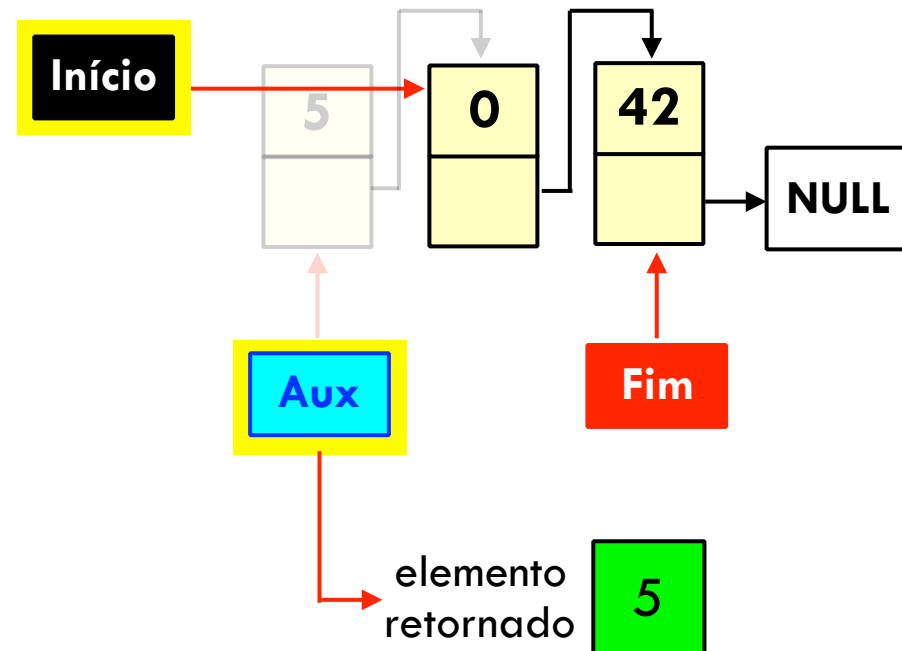
# Remoção (dequeue)

b) desenfileirar (remover elemento 42)

Número de elementos : 3



Número de elementos : 2



# Remoção (dequeue)

b) desenfileirar (remover elemento 42)

Número de elementos : 3

Início

5

0

42

O que houve?

1. criamos uma variável auxiliar **Aux** (Ponteiro)
2. **Aux** recebe o **Início** da Fila
3. **Início** recebe o próximo do **Início**
4. contador é decrementado
5. desalocamos a memória do nó removido
6. retorna o elemento armazenado em **Aux**

Número de elementos : 2

Início

5

0

42

NULL

Aux

Fim

elemento  
retornado

5

# Remoção (dequeue)

## Dequeue (Q)

1. se a fila não está vazia:
2.     **Aux** recebe o **Inicio** da Fila
3.     x recebe o valor x do Item em **Aux**
4.     **Inicio** recebe o proximo de **Inicio**
5.     desalocamos memoria de **Aux**
6.     decrementa contador de elementos
7.     **retorna (x);**

# Remoção (dequeue)

## Dequeue (Q)

1. se a fila não está vazia:
2.     Aux = Q.Inicio
3.     x = Aux.x;
4.     Q.Inicio = Q.Inicio->Proximo
5.     free(Aux)
6.     decrementa contador de elementos
7.     return(x);

# Funções adicionais?



- Quais outras funções podem ser úteis para o tipo Fila?



# Exercício 01

- Ilustre cada estado de uma fila após realizar as seguintes operações (em ordem)
  - Enqueue(Q, 8)
  - Enqueue(Q, 10)
  - Enqueue(Q, 12)
  - Dequeue(Q)
  - Enqueue(Q, -1)
  - Dequeue(Q)
  - Dequeue(q)
- Considere que a pilha está inicialmente vazia

## Exercício 02

- Mãos a obra: implemente um TDA para Fila com alocação dinâmica, e as funções de manipulação.
- Quais TDAs serão necessários?

# Tipo Abstrato para Fila Estática

```
typedef struct {  
    int key;  
} Item;  
  
typedef struct NoFila *Ponteiro;  
  
typedef struct NoFila {  
    Item topo;  
    Ponteiro proximo;  
} NoFila;  
  
typedef struct {  
    Ponteiro inicio;  
    Ponteiro fim;  
    int tamanho;  
} FilaDinamica;
```

# Tipo Abstrato para Fila Estática

```
typedef struct {  
    int key;  
} Item;  
  
typedef struct NoFila *Ponteiro;  
  
typedef struct NoFila {  
    Item topo;  
    Ponteiro proximo;  
} NoFila;  
  
typedef struct {  
    Ponteiro inicio;  
    Ponteiro fim;  
    int tamanho;  
} FilaDinamica;
```

# Tipo Abstrato para Fila Estática

```
typedef struct {  
    int key;  
} Item;
```

.....▶ implementa o nosso  
objeto

```
typedef struct NoFila *Ponteiro;
```

.....▶ implementa o tipo que permite  
concatenar os nós dinâmicos

```
typedef struct NoFila {  
    Item topo;  
    Ponteiro proximo;  
} NoFila;
```

.....▶ implementa os nós da fila  
(estrutura recursiva) !!!

```
typedef struct {  
    Ponteiro inicio;  
    Ponteiro fim;  
    int tamanho;  
} FilaDinamica;
```

.....▶ implementa o TDA  
para Fila

# Implementação (Dinâmica)

```
void iniciaFila(FilaDinamica *fila);  
void enqueue(Item item, FilaDinamica *fila);  
void dequeue(FilaDinamica *fila, Item *item);  
void imprimeFila(FilaDinamica *fila);  
int estaVazia(FilaDinamica *fila);  
int tamanhoFila(FilaDinamica *fila);  
Item primeiro(FilaDinamica *fila);  
Item ultimo(FilaDinamica *fila);
```

# Próximas Aulas



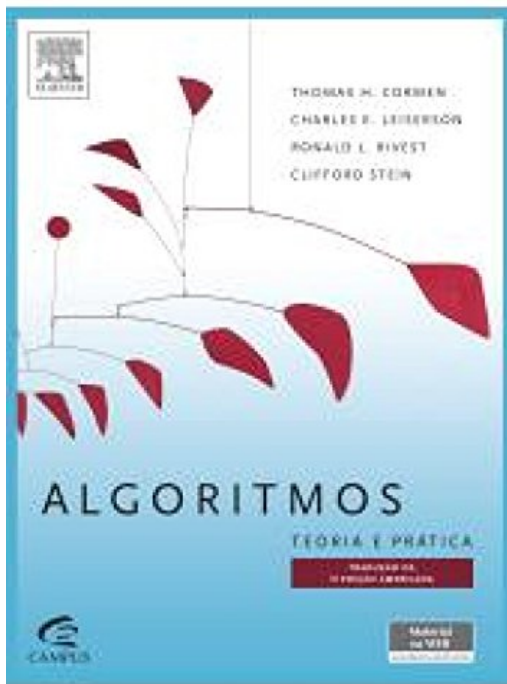
- Listas Lineares
  - single-linked
  - double-linked

# Roteiro

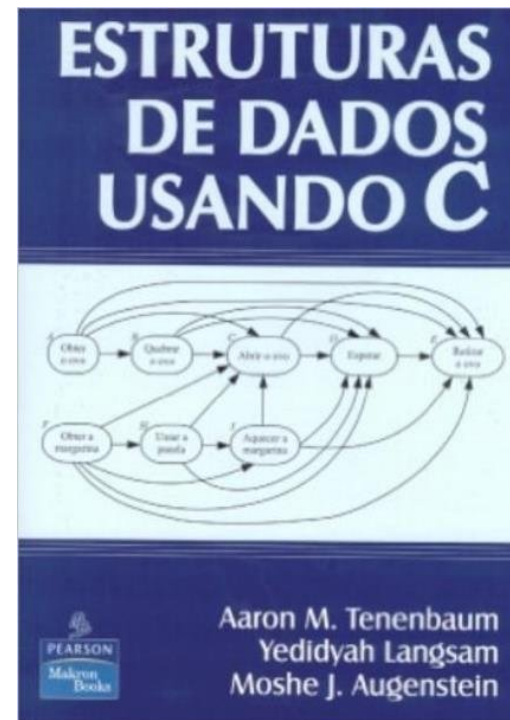
- 1 Introdução
- 2 Filas
- 3 Operações gerais
- 4 Inserção de elementos
- 5 Remoção de elementos
- 6 Referências



# Referências sugeridas

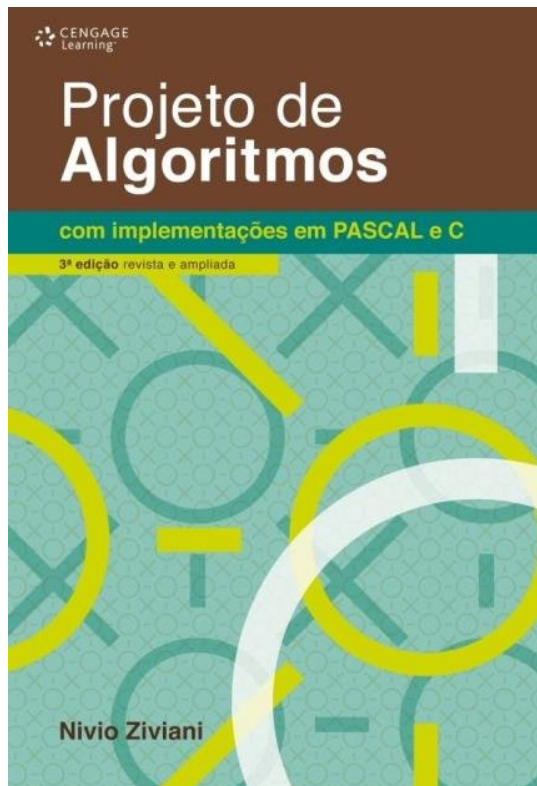


[Cormen et al, 2018]



[Tenenbaum et al, 1995]

# Referências sugeridas



[Ziviani, 2010]



[Drozdek, 2017]

# Perguntas?

Prof. Rafael G. **Mantovani**

[rafaelmantovani@utfpr.edu.br](mailto:rafaelmantovani@utfpr.edu.br)