

# Algoritmos e Estruturas de Dados I

## Aula 05: Filas

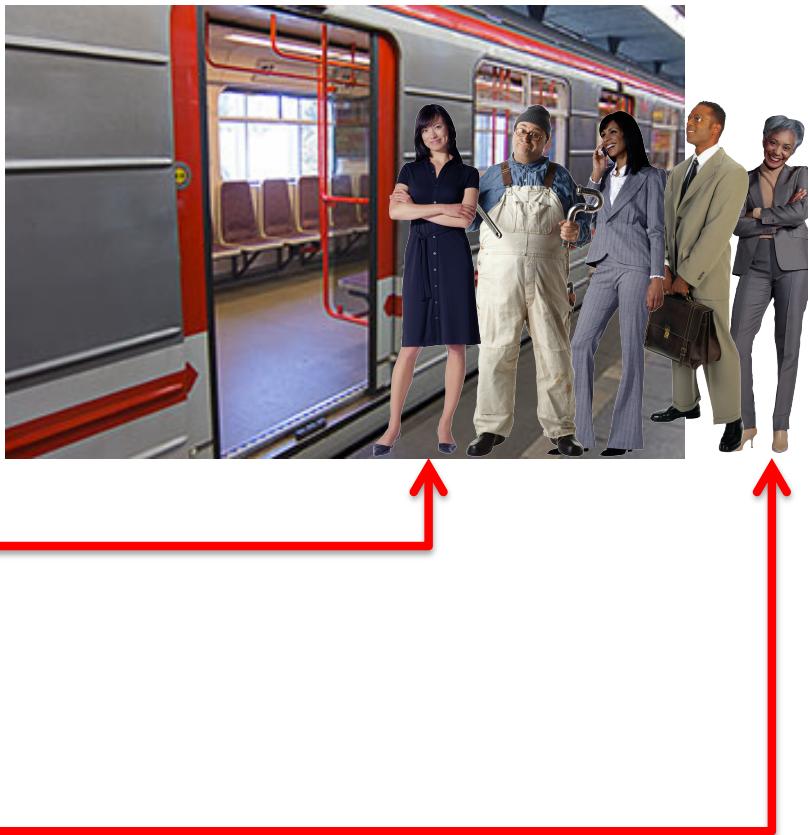
Prof. Márcio Porto Basgalupp

*créditos: Prof. Jurandy G. Almeida Jr.*

Universidade Federal de São Paulo  
Departamento de Ciência e Tecnologia

# O que é uma fila?

- Uma **fila** é uma estrutura de dados na qual todas as retiradas e, em geral, os acessos são realizados em uma extremidade (frente), e todas as inserções são efetuadas em outra extremidade (trás)



# O que é uma fila?

- Por exemplo, pense numa fila de pessoas para pegar o metrô
- As pessoas vão chegando ...



# O que é uma fila?

- Por exemplo, pense numa fila de pessoas para pegar o metrô
- As pessoas vão chegando ...



# O que é uma fila?

- Por exemplo, pense numa fila de pessoas para pegar o metrô
- As pessoas vão chegando ...



# O que é uma fila?

- Por exemplo, pense numa fila de pessoas para pegar o metrô
- As pessoas vão chegando ...



# O que é uma fila?

- Por exemplo, pense numa fila de pessoas para pegar o metrô
- As pessoas vão chegando uma após a outra



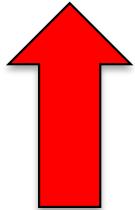
# O que é uma fila?

- A primeira pessoa a chegar é a primeira a entrar no metrô
- A última pessoa a chegar é a última a entrar no metrô



# Operações Básicas

- Quando um item é adicionado numa fila, usa-se a operação **Enfileirar** (inserir na parte de trás da fila)



# Operações Básicas

- Quando um item é adicionado numa fila, usa-se a operação **Enfileirar** (inserir na parte de trás da fila)



# Operações Básicas

- Quando um item é adicionado numa fila, usa-se a operação **Enfileirar** (inserir na parte de trás da fila)
- Quando um item é retirado de uma fila, usa-se a operação **Desenfileirar** (retirar da parte da frente da fila)



# Operações Básicas

- Quando um item é adicionado numa fila, usa-se a operação **Enfileirar** (inserir na parte de trás da fila)
- Quando um item é retirado de uma fila, usa-se a operação **Desenfileirar** (retirar da parte da frente da fila)



# Operações Básicas

- Quando um item é adicionado numa fila, usa-se a operação **Enfileirar** (inserir na parte de trás da fila)
- Quando um item é retirado de uma fila, usa-se a operação **Desenfileirar** (retirar da parte da frente da fila)



# Propriedades

- O **primeiro** item inserido na fila é sempre o **primeiro** a ser retirado
- Essa propriedade é denominada *First In, First Out* ou **FIFO**
- Existe uma ordem linear que é a **ordem de chegada**



# Exemplo

- fila inicialmente vazia

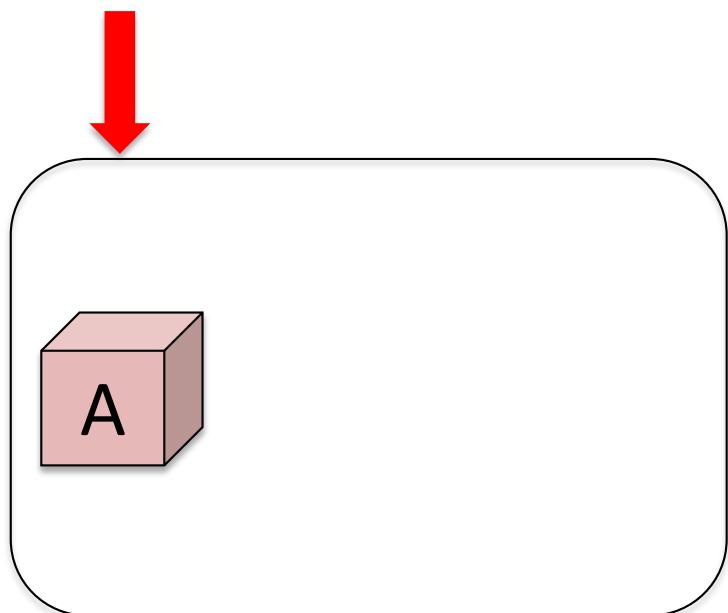
trás  
frente



# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A

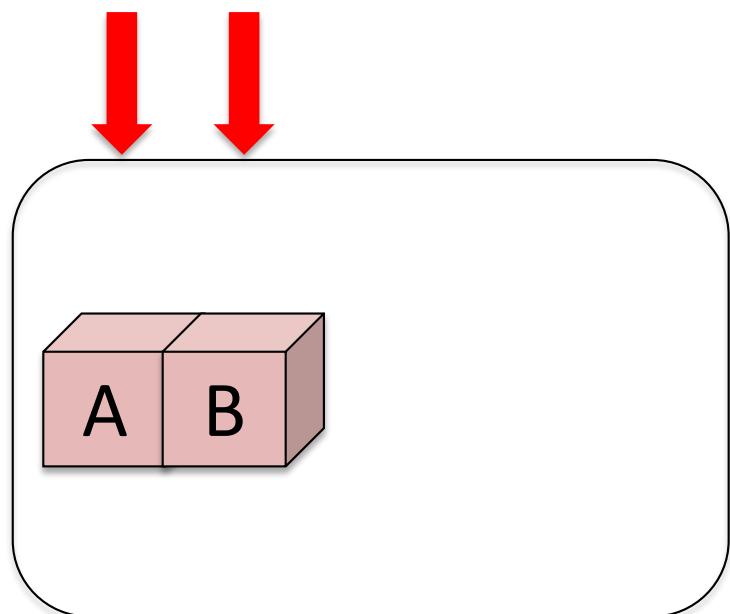
trás  
frente



# Exemplo

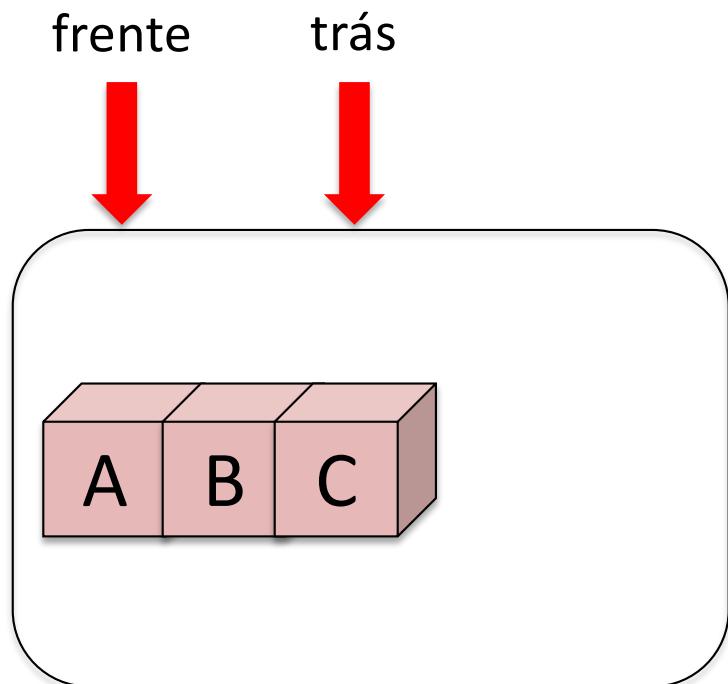
- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B

frente trás



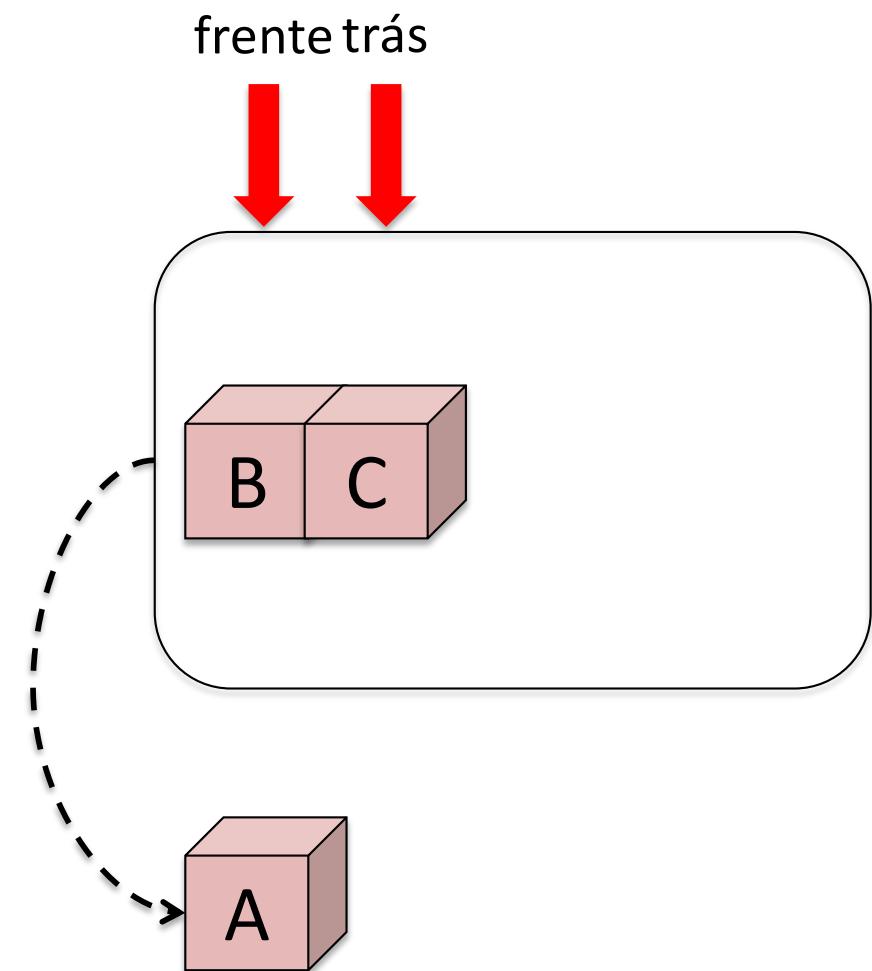
# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B
- inserir (enfileirar) item C



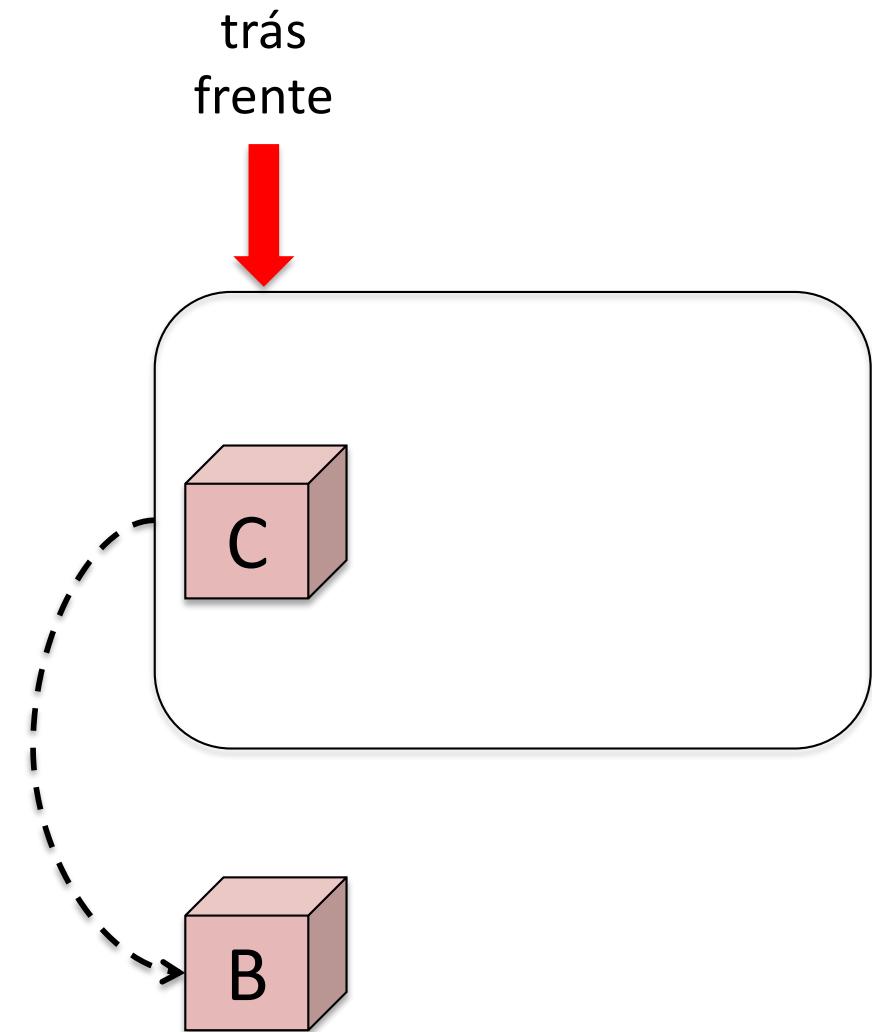
# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B
- inserir (enfileirar) item C
- retirar (desenfileirar) item



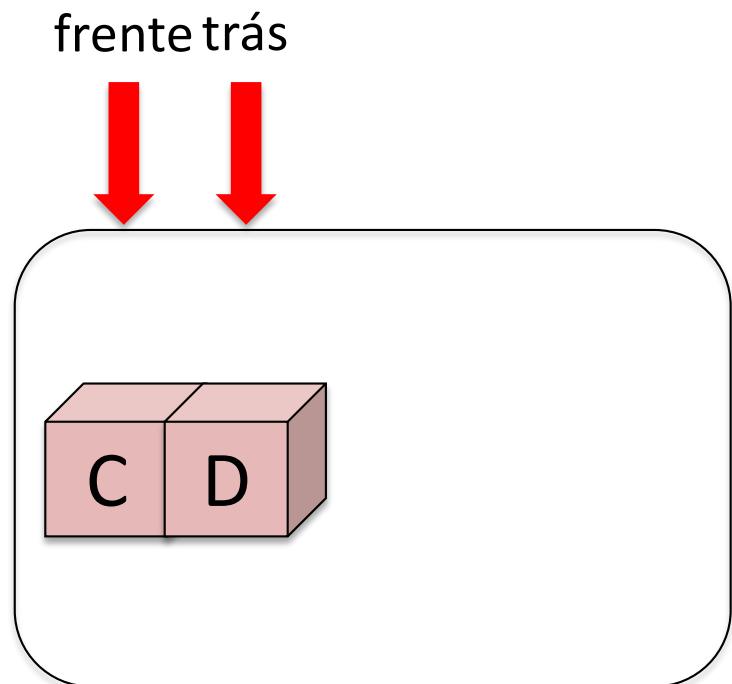
# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B
- inserir (enfileirar) item C
- retirar (desenfileirar) item
- retirar (desenfileirar) item



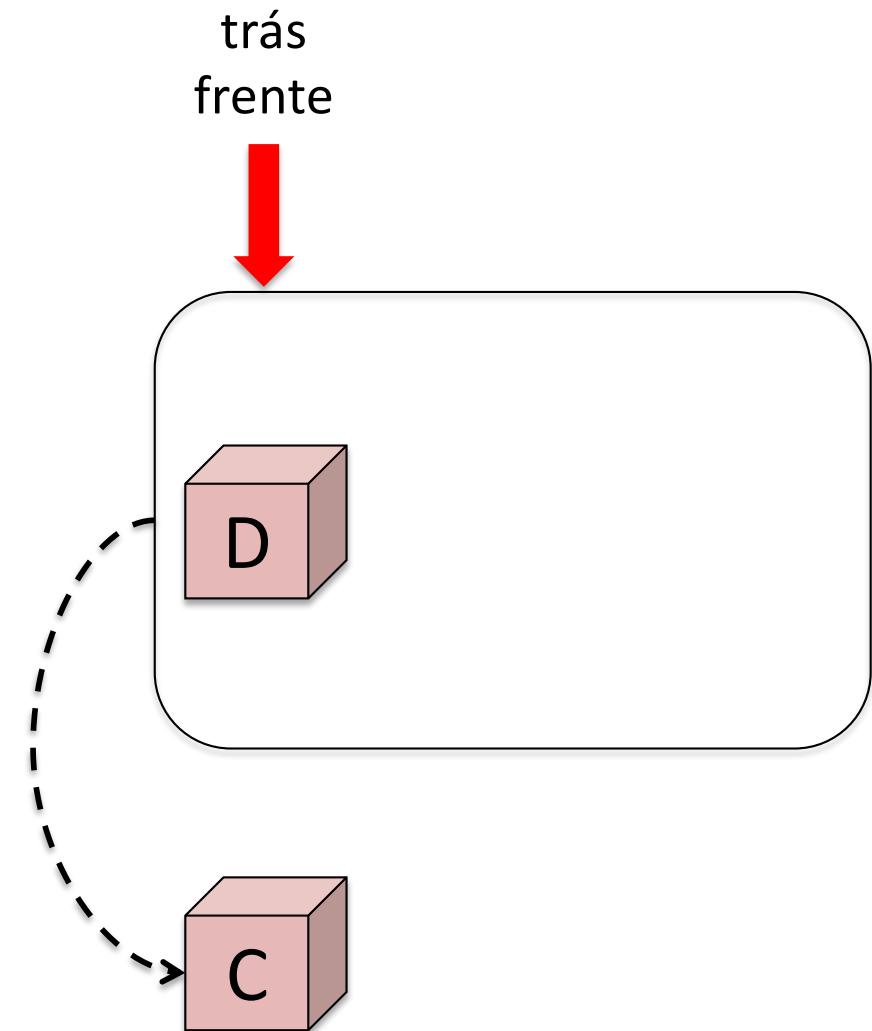
# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B
- inserir (enfileirar) item C
- retirar (desenfileirar) item
- retirar (desenfileirar) item
- inserir (enfileirar) item D



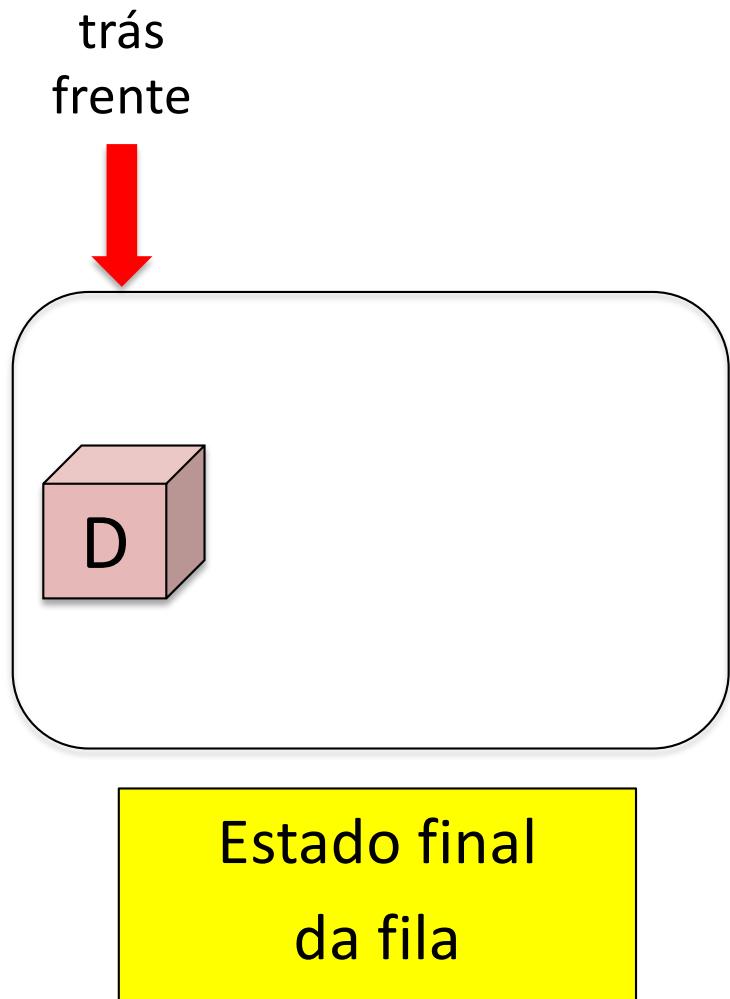
# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B
- inserir (enfileirar) item C
- retirar (desenfileirar) item
- retirar (desenfileirar) item
- inserir (enfileirar) item D
- retirar (desenfileirar) item



# Exemplo

- fila inicialmente vazia
- inserir (enfileirar) item A
- inserir (enfileirar) item B
- inserir (enfileirar) item C
- retirar (desenfileirar) item
- retirar (desenfileirar) item
- inserir (enfileirar) item D
- retirar (desenfileirar) item



- É ideal para aplicações que operam de acordo com a ordem “*primeiro-que-chega, primeiro-atendido*”
  - Ex: sistemas operacionais
    - Filas de impressão, processamento, ...

## ■ Conjunto de Operações

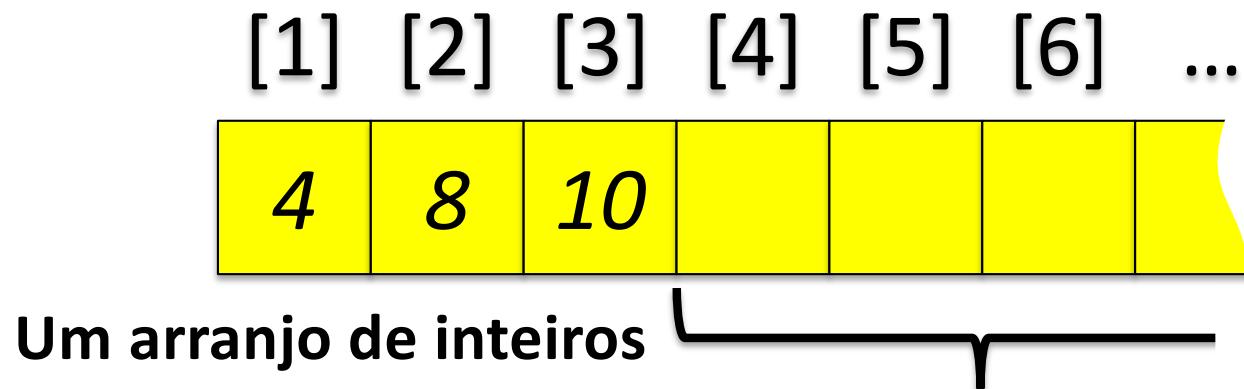
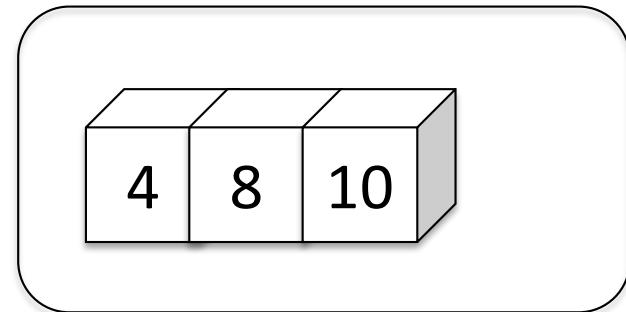
- **TFila\_Inicia(Fila)**: Inicia uma fila vazia
- **TFila\_EhVazia(Fila)**: Retorna *true* se a fila estiver vazia; caso contrário, retorna *false*
- **TFila\_Enfileira(Fila, x)**: Insere o item *x* na parte de trás da fila
- **TFila\_Desenfileira(Fila, x)**: Retorna o item *x* da parte da frente da fila, retirando-o da fila
- **TFila\_Tamanho(Fila)**: Retorna o número de itens da fila

- Existem várias opções de estruturas de dados que podem ser usadas para representar filas
- As duas representações mais utilizadas são:
  - Implementação por meio de **arranjos**
  - Implementação por meio de **apontadores**

# IMPLEMENTAÇÃO POR ARRANJOS

# Implementação por Arranjos

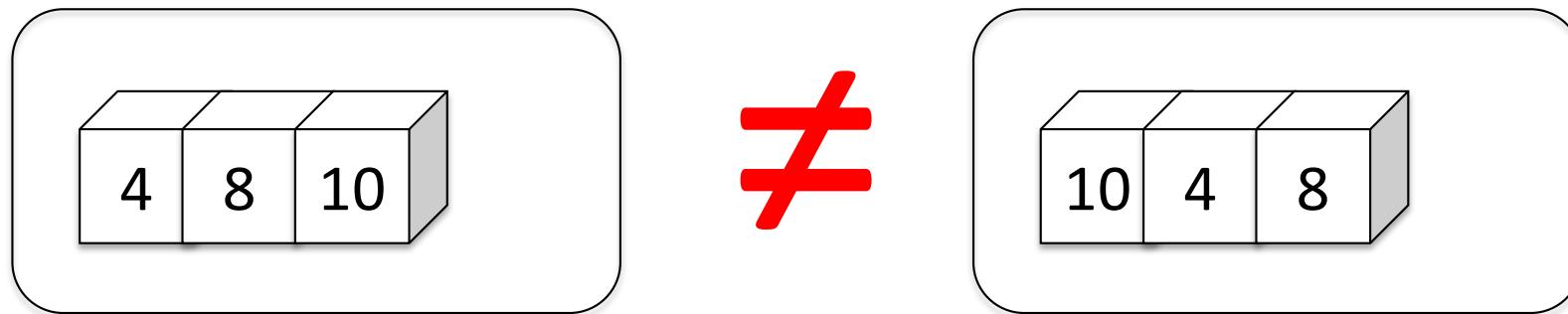
- Os itens da fila serão inicialmente armazenados no **início de um arranjo**, como mostra este exemplo



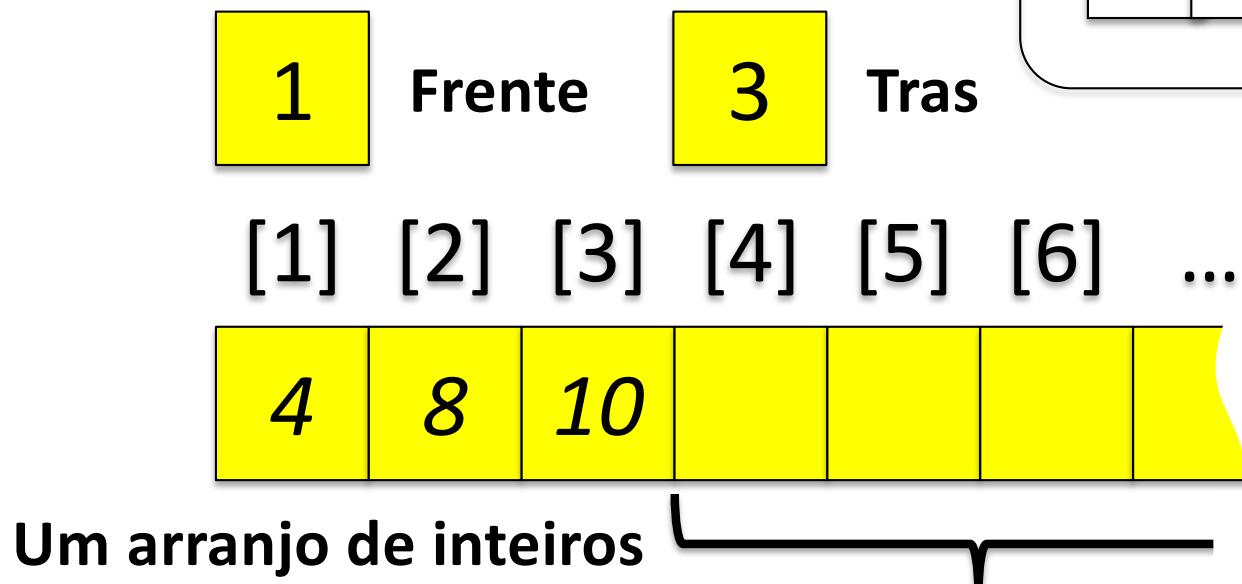
Não nos interessamos para o que está armazenado nesta parte do arranjo

# Implementação por Arranjos

- Assim, ...



- Nós precisamos também armazenar apontadores que indicam **Frente** e **Tras** da fila



Não nos interessamos para o que está armazenado nesta parte do arranjo

- Os itens são armazenados em um **arranjo** de tamanho suficiente para armazenar a fila
- O campo **Frente** mantém um apontador para o primeiro item da fila
- O campo **Tras** mantém um apontador para o último item da fila
- A constante **MaxTam** define o tamanho máximo permitido para a fila

# Estrutura da Fila por Arranjos

```
#define MAXTAM 1000

typedef int TChave;

typedef struct {
    TChave Chave;
    /* outros componentes */
} TItem;

typedef int TApontador;

typedef struct {
    TItem Item[MAXTAM];
    TApontador Frente, Tras;
} TFila;
```

# Estrutura da Fila por Arranjos

```
/* procedimentos e funções do TAD */

void TFila_Inicia(TFila *pFila);
int TFila_EhVazia(TFila *pFila);
int TFila_Enfileira(TFila *pFila, TItem x);
int TFila_Desenfileira(TFila *pFila, TItem *pX);
int TFila_Tamanho(TFila *pFila);
```

- Os itens da fila são armazenados em posições contíguas de memória
- A fila tende a caminhar pela memória, ocupando espaço na parte de trás e descartando espaço na parte da frente
  - A operação **TFila\_Enfileira** faz a parte de trás da fila expandir-se
  - A operação **TFila\_Desenfileira** faz a parte da frente da fila contrair-se

## ■ Problema:

- Com poucas inserções e retiradas, a fila vai ao encontro do limite do espaço de memória alocado para ela

## ■ Soluções:

- Modelo físico,
- Arranjo com dois índices
- Arranjo circular

## ■ Modelo físico

- Um arranjo com início sempre na primeira posição e todas as entradas movidas quando um item é removido
  - Método lento e pobre para ser usado em computadores

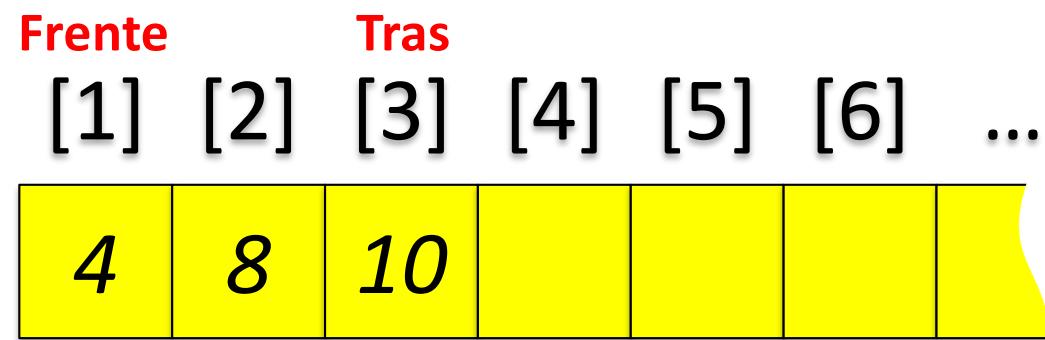
## ■ Arranjo com dois índices

- Frente e trás que sempre crescem
  - Útil se a fila pode ser esvaziada totalmente quando cheia

## ■ Arranjo circular

- O arranjo é visto como um círculo
  - A primeira posição segue a última

- Fila antes de uma retirada



- Fila antes de uma retirada

Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
		4	8	10				

- Fila depois de uma retirada

x	Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
4			4	8	10				

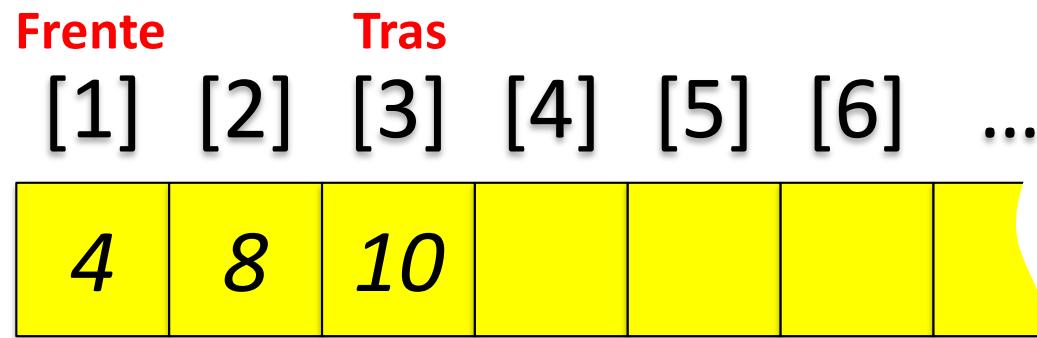
- Fila antes de uma retirada

Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
		4	8	10				

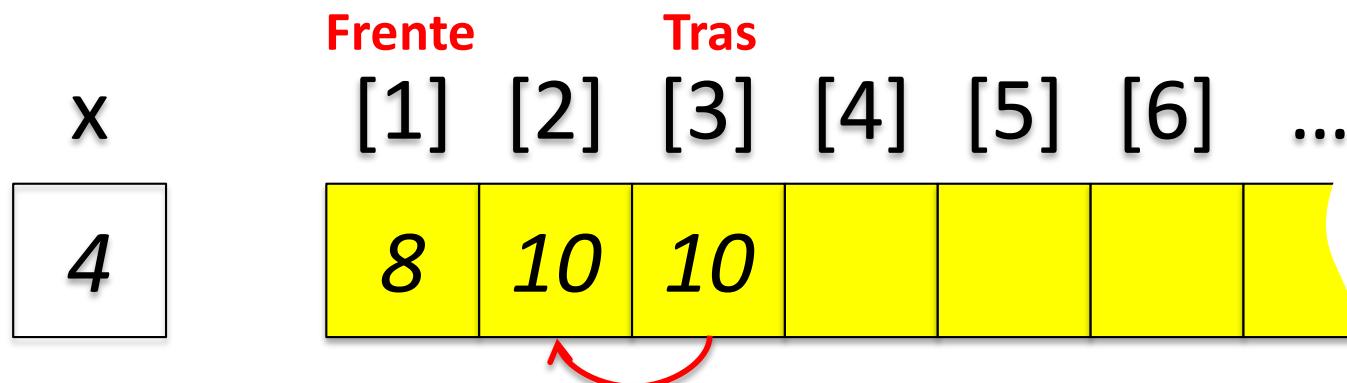
- Fila depois de uma retirada

x	Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
4			8	8	10				

- Fila antes de uma retirada



- Fila depois de uma retirada



- Fila antes de uma retirada

Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
		4	8	10				

- Fila depois de uma retirada

x	Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
4			8	10					

# Arranjo com Dois Índices

- Fila antes de uma retirada

Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
4	8	10						

# Arranjo com Dois Índices

- Fila antes de uma retirada

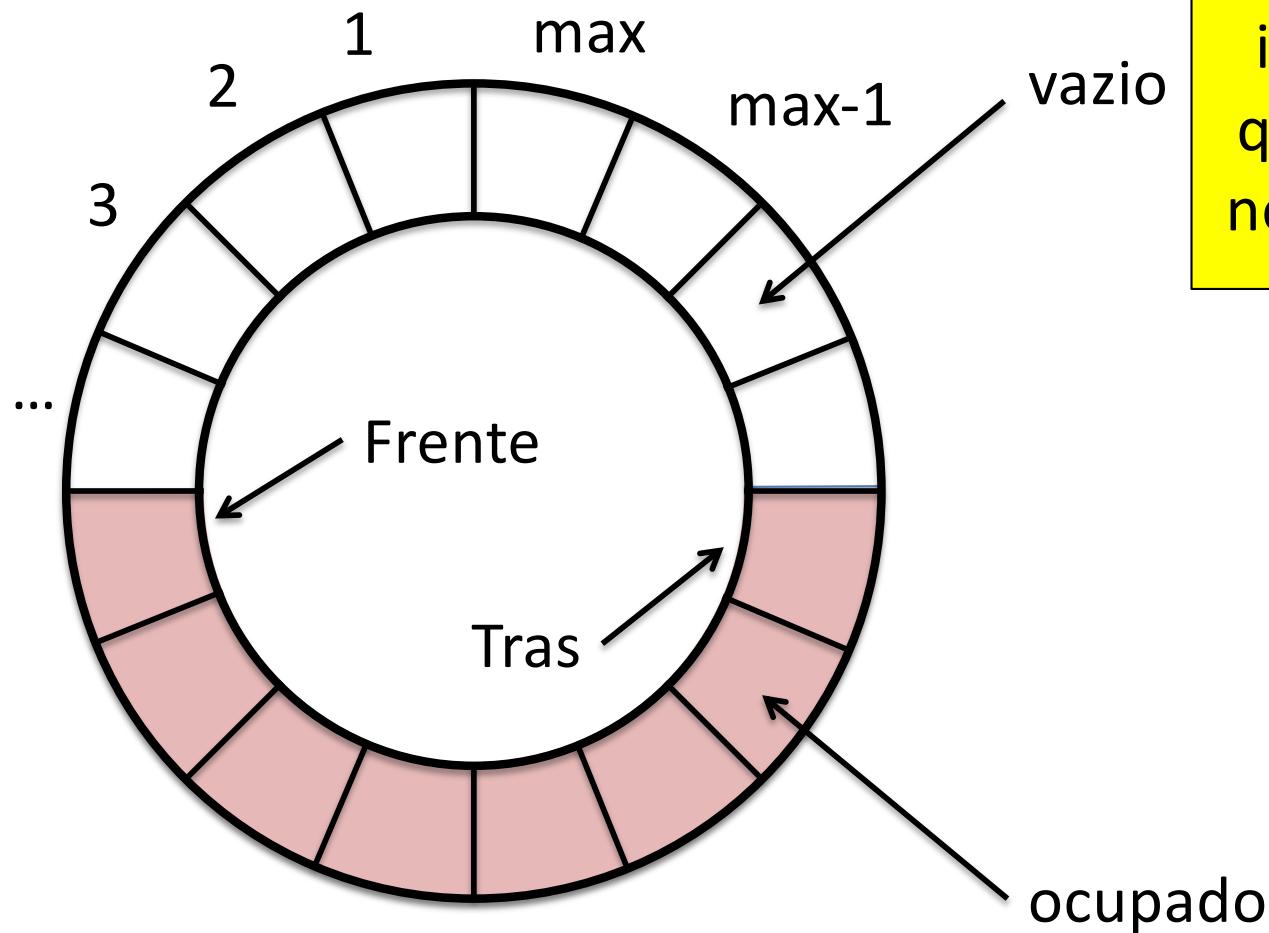
Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
		4	8	10				

- Fila depois de uma retirada

x	Frente	Tras	[1]	[2]	[3]	[4]	[5]	[6]	...
4				8	10				

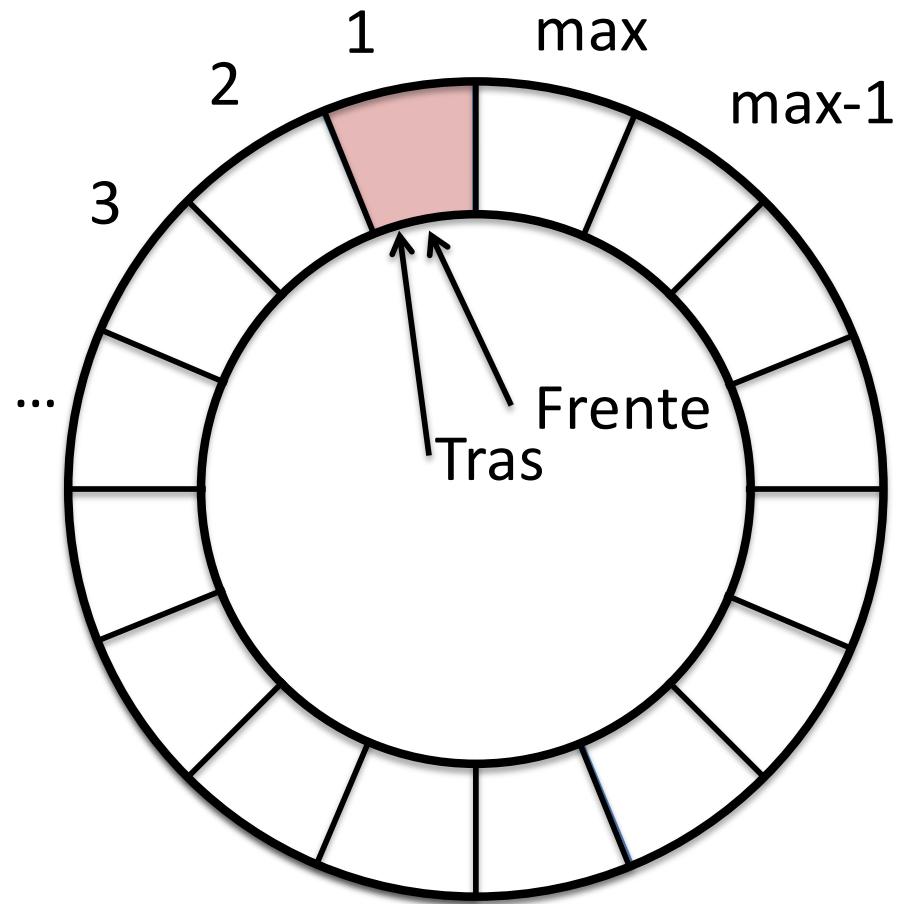
- Para implementar um arranjo circular, suponha um círculo numerado de 1 até **MaxTam**
  - **MaxTam** indica o número de elementos do arranjo circular
- O movimento de índices **Frente** e **Tras** é dado pelo módulo aritmético: quando um índice ultrapassa **MaxTam**, ele recomeça novamente em 1
  - Similar a um relógio, em que as horas são numeradas de 1 a 12 e ao adicionar 4 horas às 10 horas, obtém-se 2 horas

# Arranjo Circular



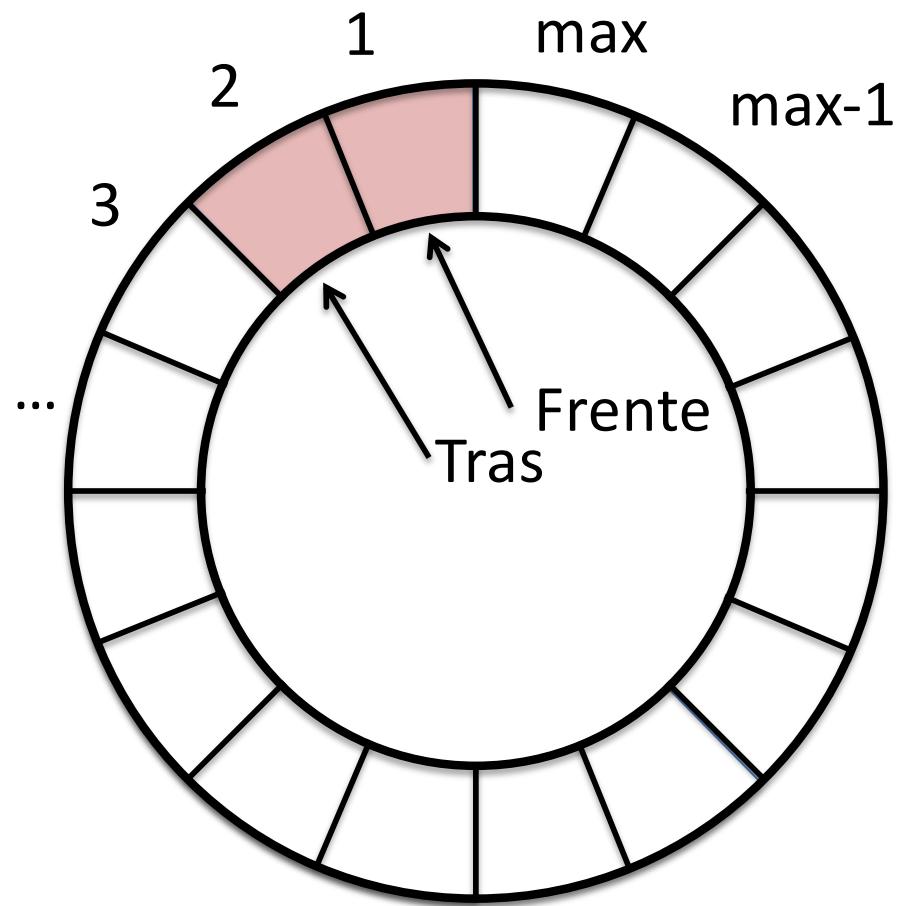
Nós não nos interessamos para o que está armazenado nesta parte do arranjo

# Arranjo Circular



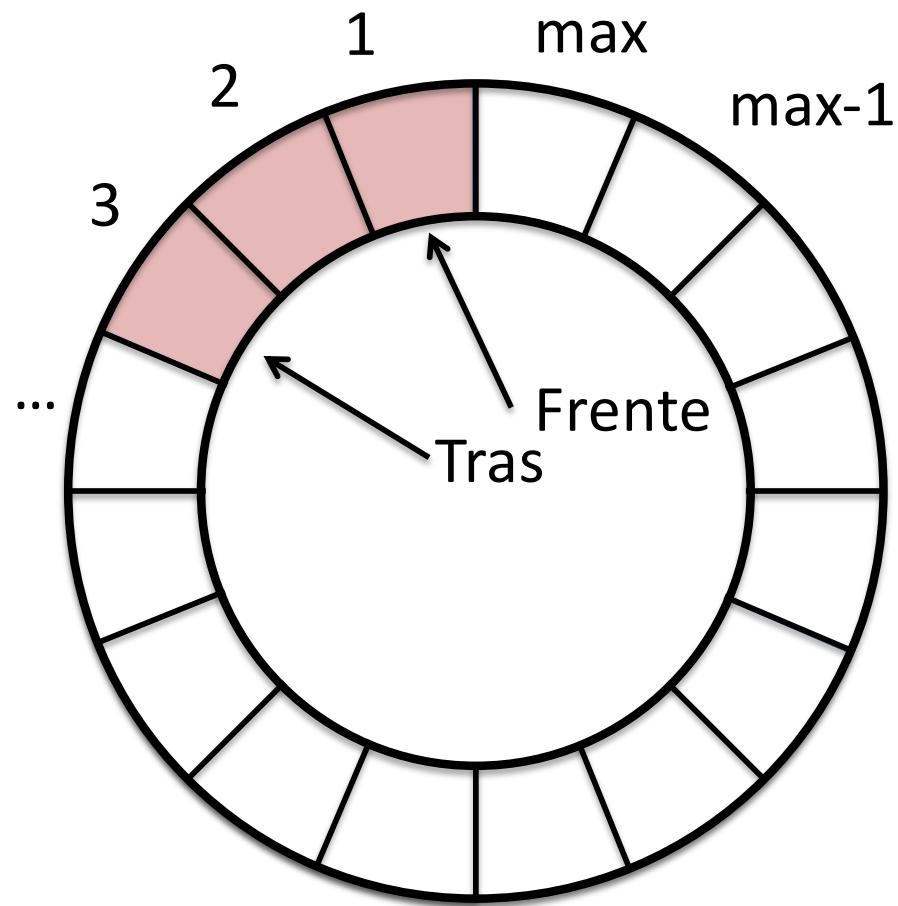
- Inserindo um item ...

# Arranjo Circular



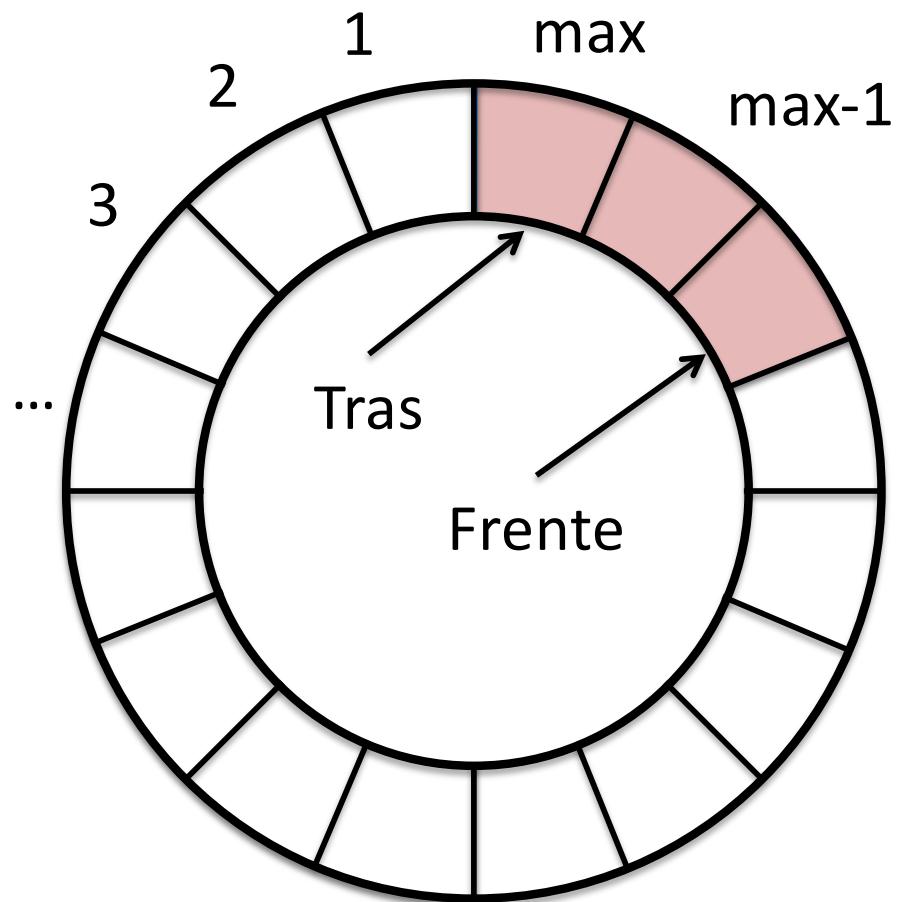
- Inserindo um item ...
- mais um ...

# Arranjo Circular



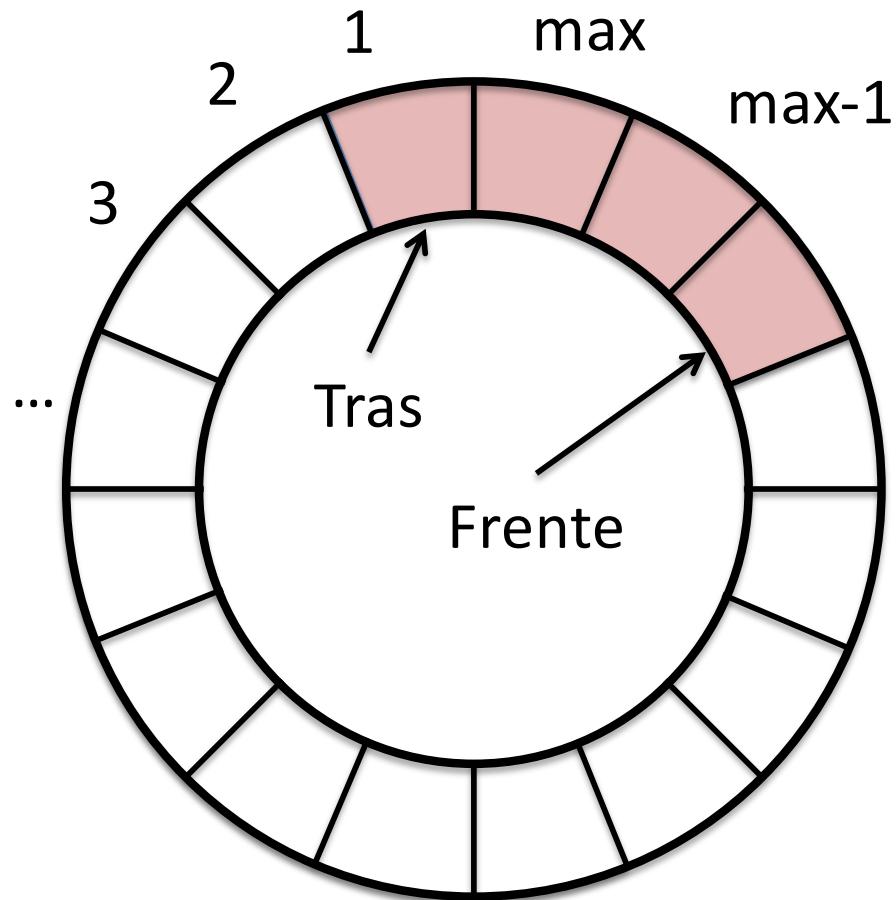
- Inserindo um item ...
- mais um ...
- e mais um ...

# Arranjo Circular



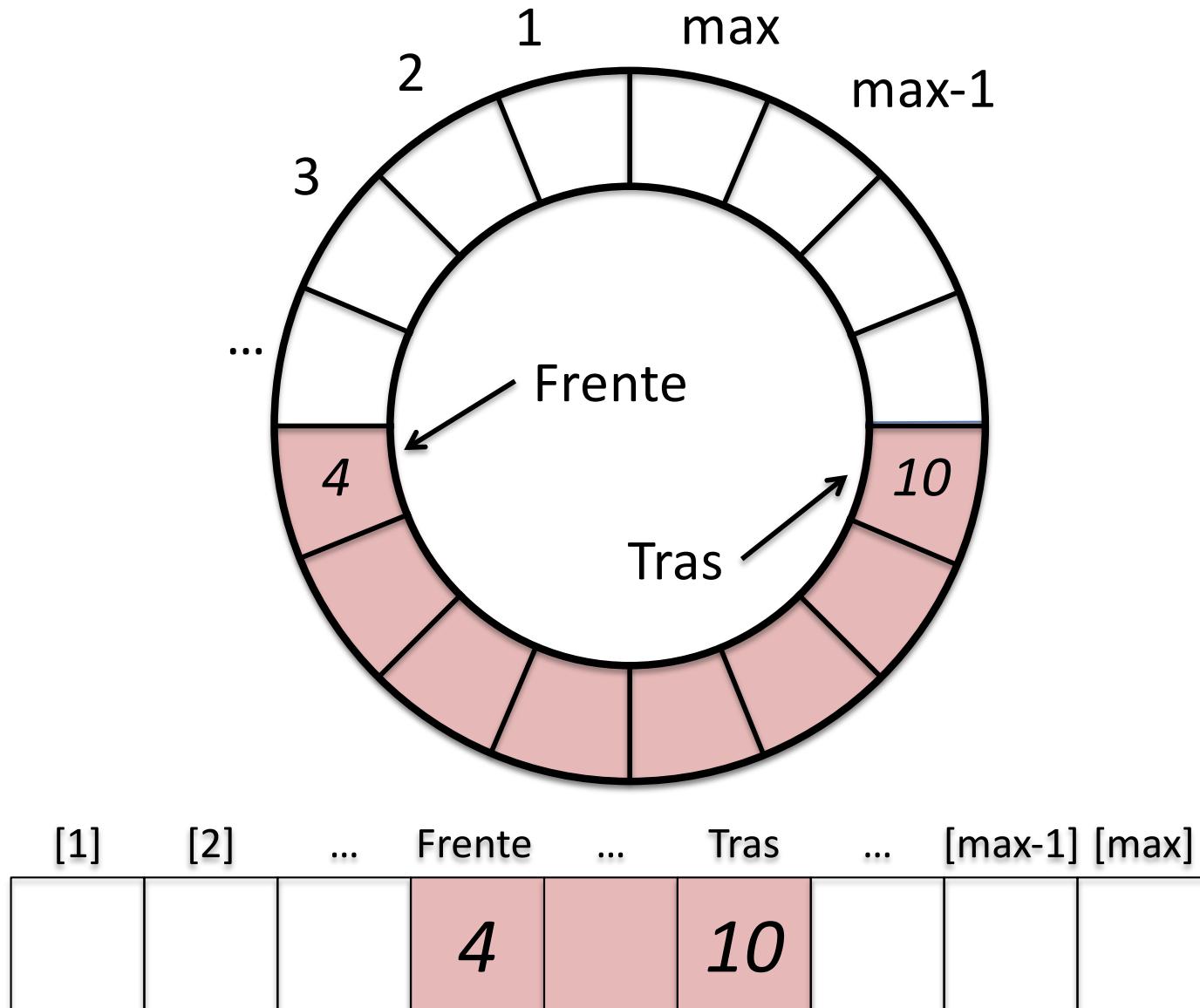
- Depois de várias inserções e remoções, chega-se ao final do arranjo ...

# Arranjo Circular



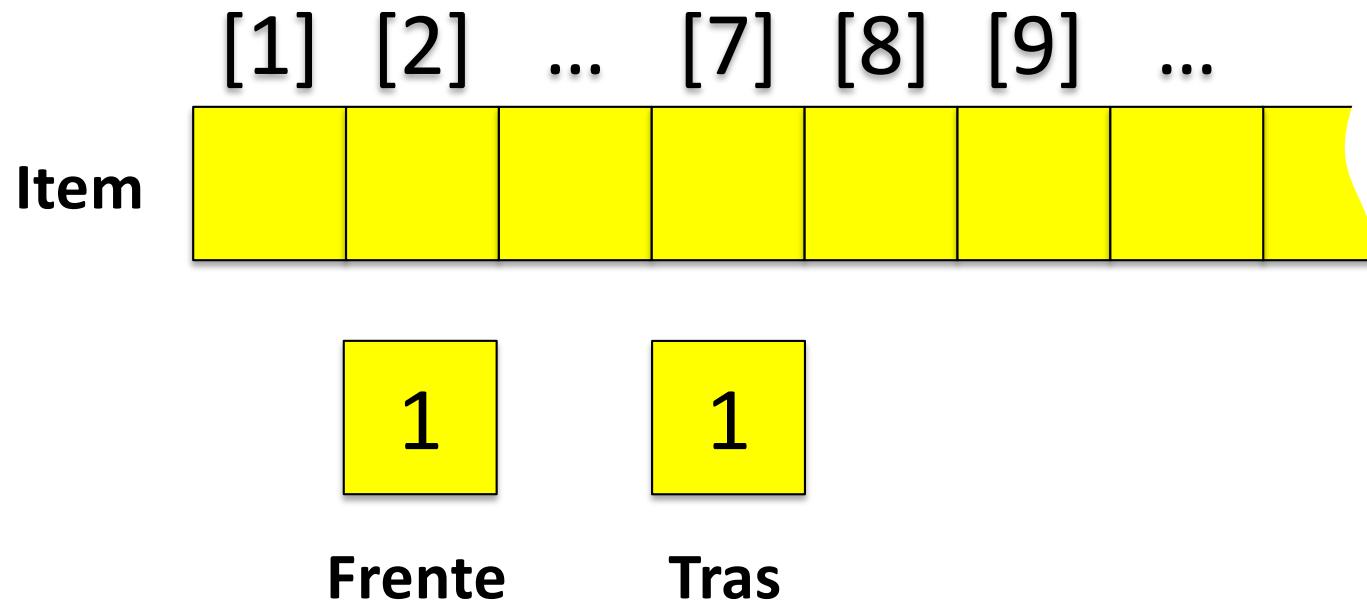
- Depois de várias inserções e remoções, chega-se ao final do arranjo ...
- ... e recomeça-se da posição 1

# Arranjo Circular

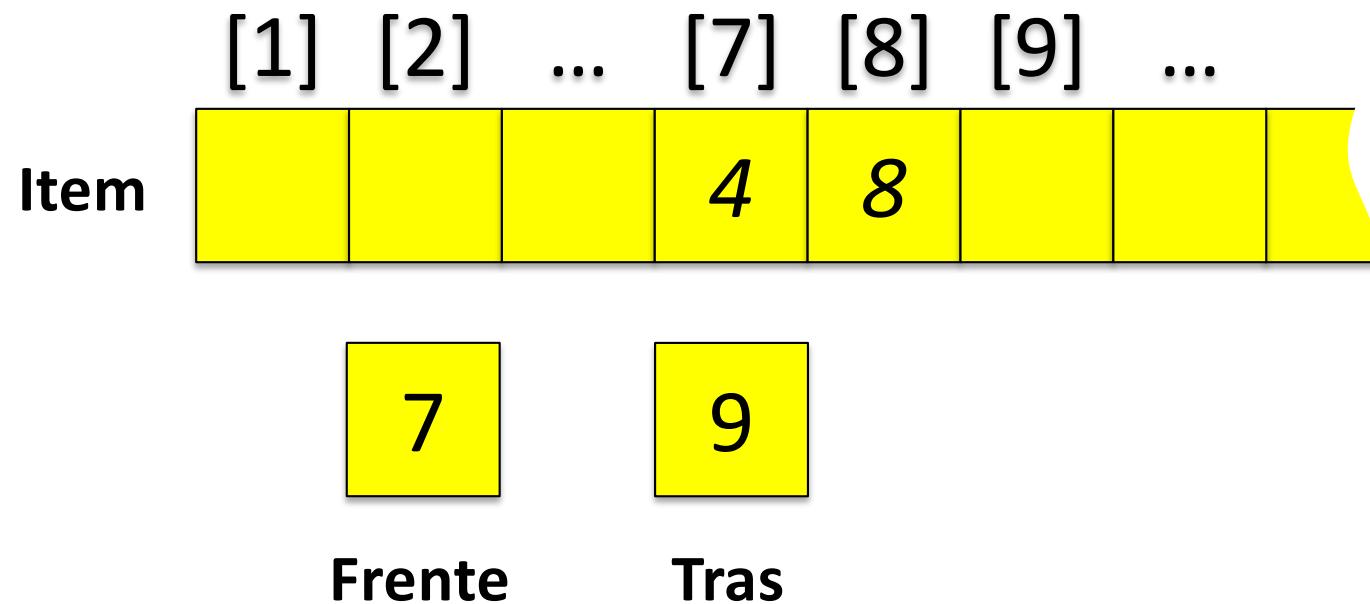


- Nos casos de fila cheia e vazia, os apontadores **Frente** e **Tras** apontam para a mesma posição do círculo
- Existem três opções para distinguir as duas situações:
  - Arranjo circular com dois índices (**Frente** e **Tras**) e uma posição sempre deixada vaga
  - Arranjo circular com dois índices (**Frente** e **Tras**) e uma variável (**Tamanho**) contendo o número de itens
  - Arranjo circular com dois índices (**Frente** e **Tras**) que assumem valores especiais para indicar fila vazia

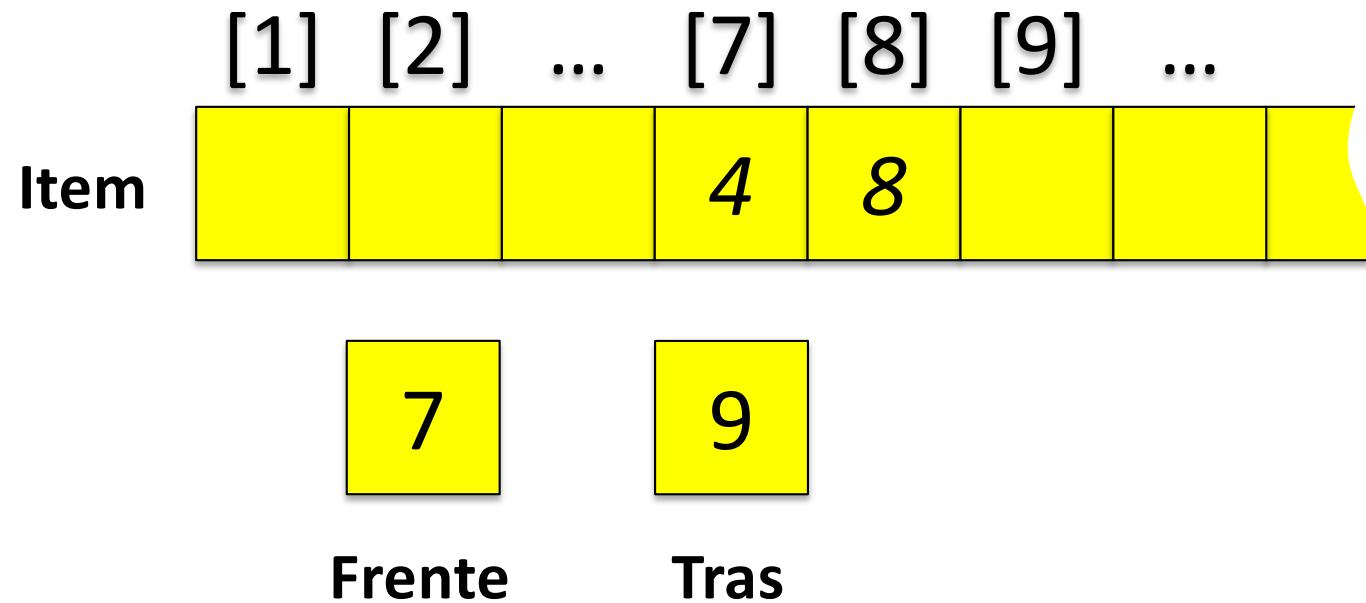
- Nos casos de fila cheia e vazia, os apontadores **Frente** e **Tras** apontam para a mesma posição do círculo
- Existem três opções para distinguir as duas situações:
  - Arranjo circular com dois índices (**Frente** e **Tras**) e uma posição sempre deixada vaga
  - Arranjo circular com dois índices (**Frente** e **Tras**) e uma variável (**Tamanho**) contendo o número de itens
  - Arranjo circular com dois índices (**Frente** e **Tras**) que assumem valores especiais para indicar fila vazia



# Inserção de Elementos na Fila

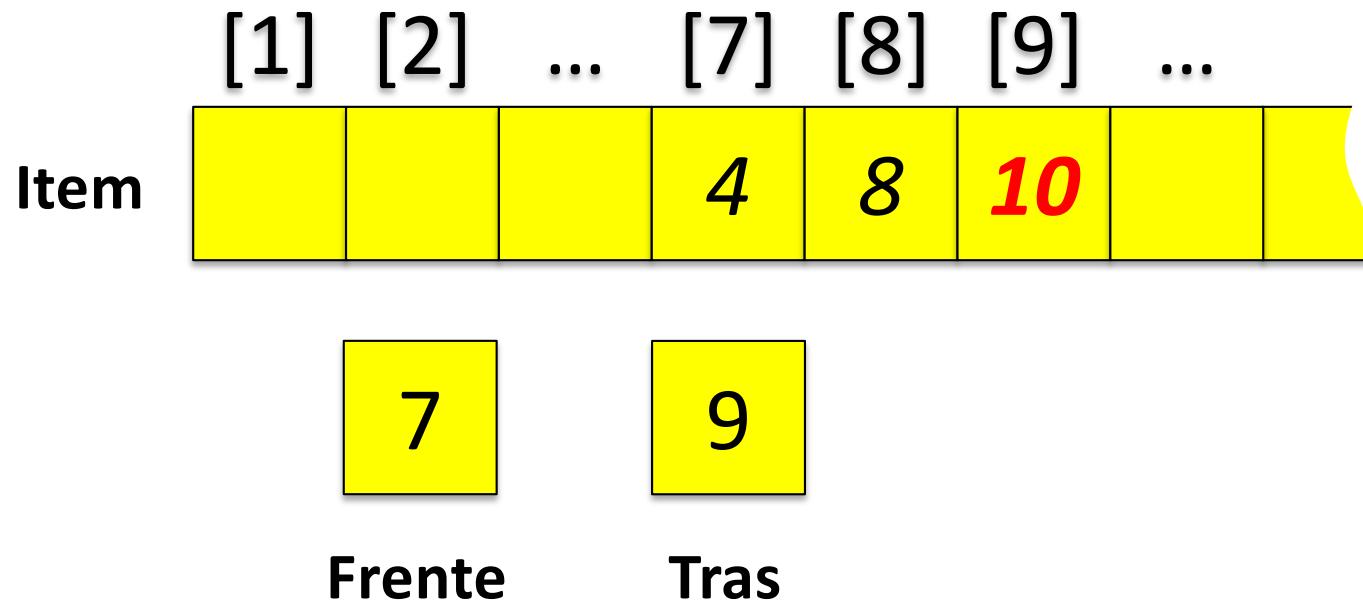


# Inserção de Elementos na Fila

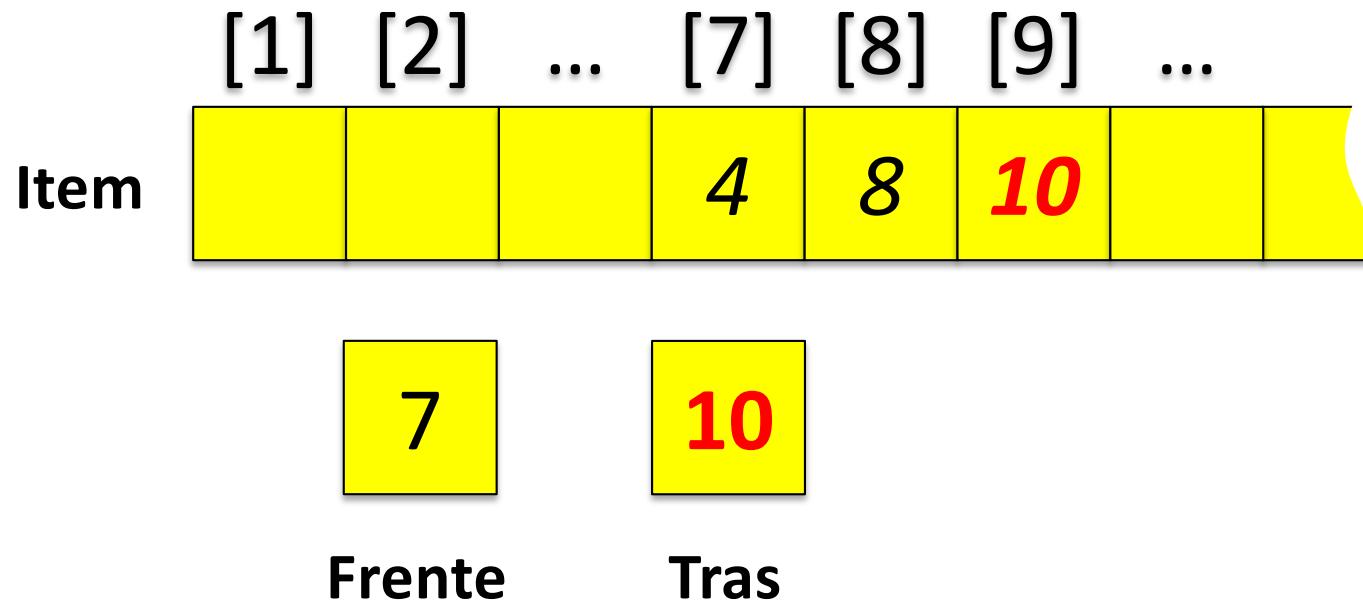


A fila está cheia?

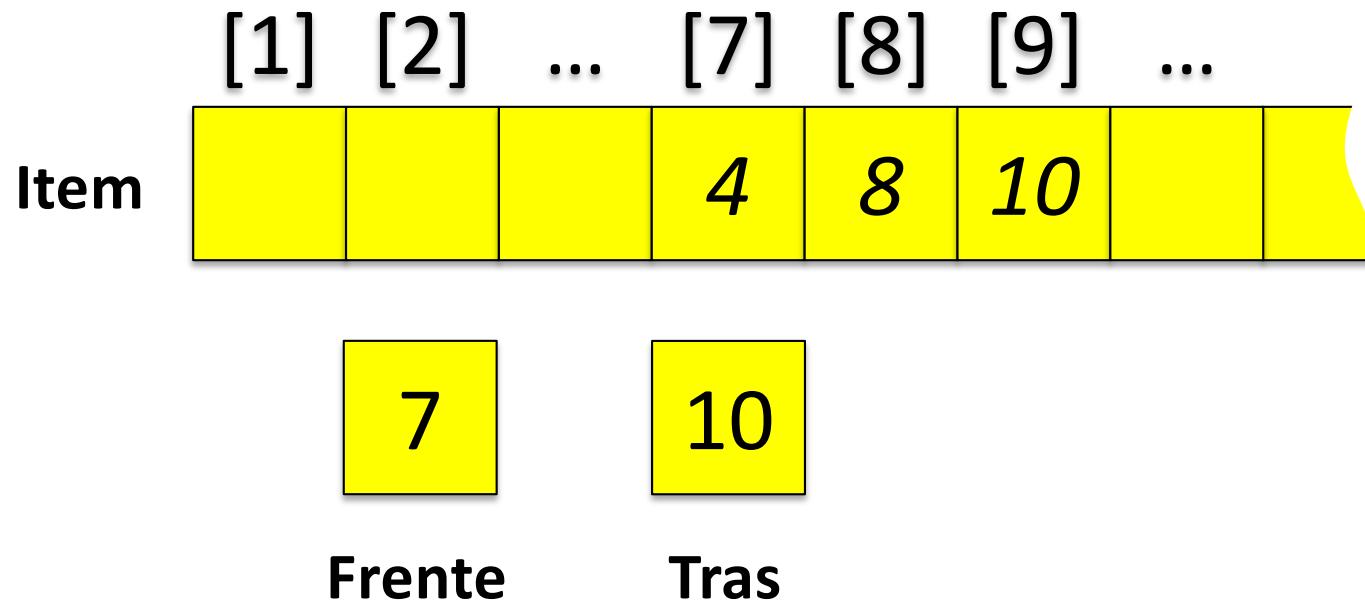
# Inserção de Elementos na Fila



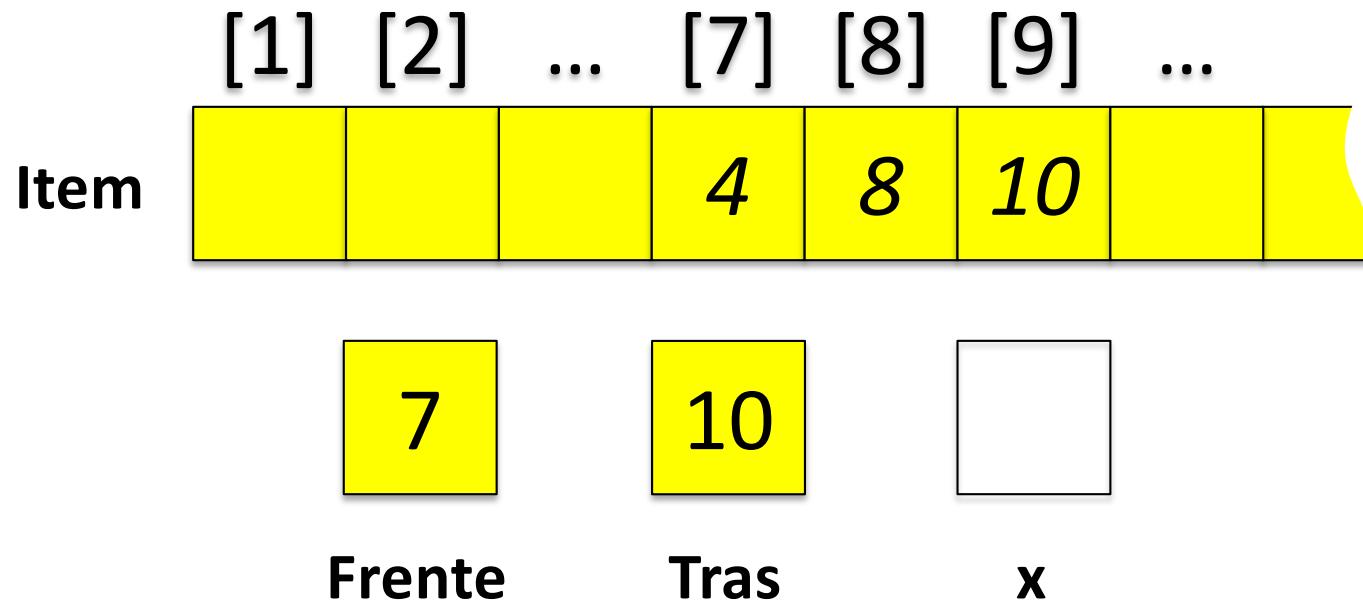
# Inserção de Elementos na Fila



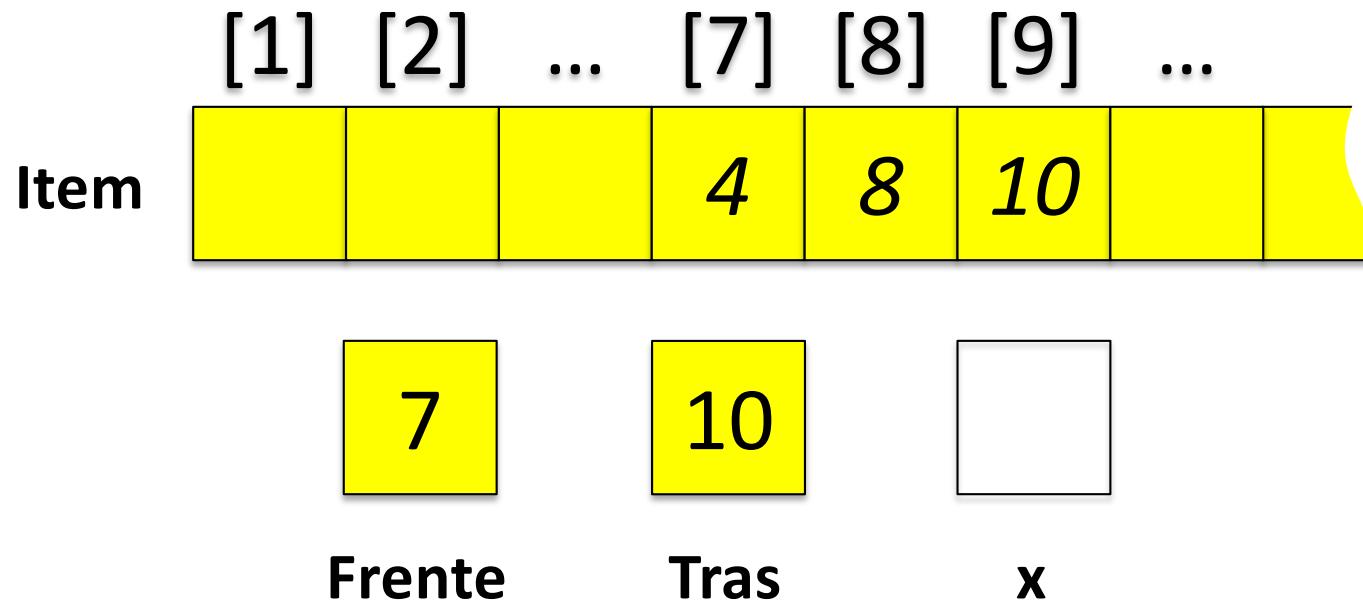
# Inserção de Elementos na Fila



# Retirada de Elementos na Fila

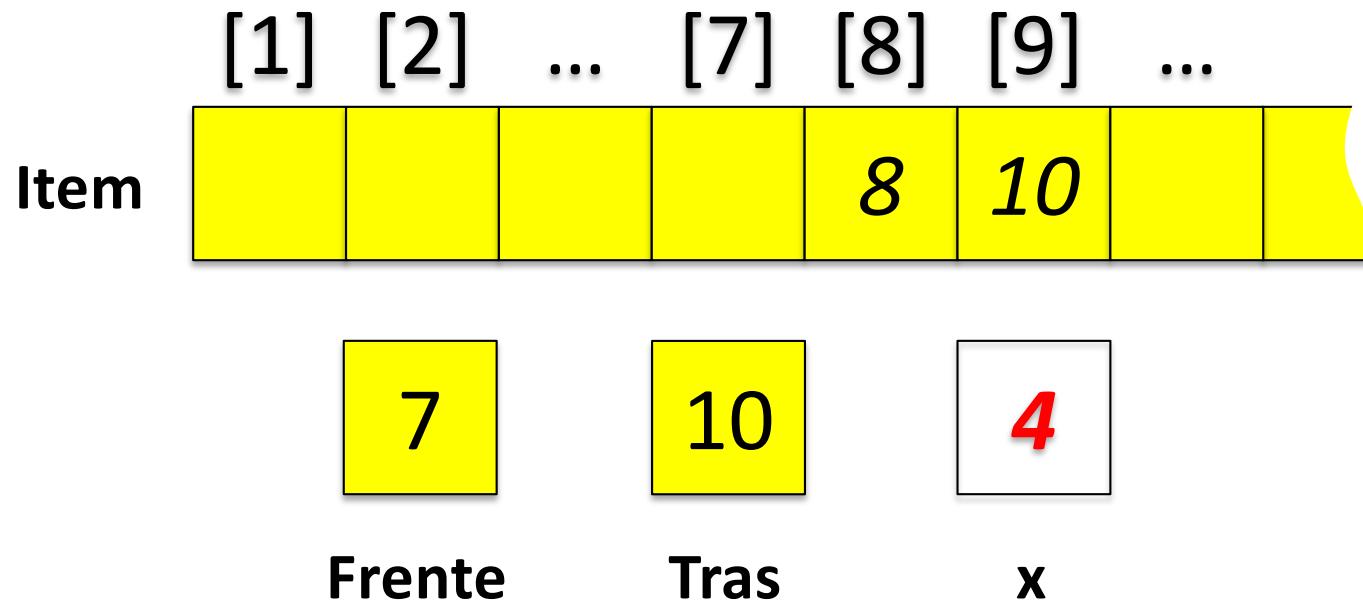


# Retirada de Elementos na Fila

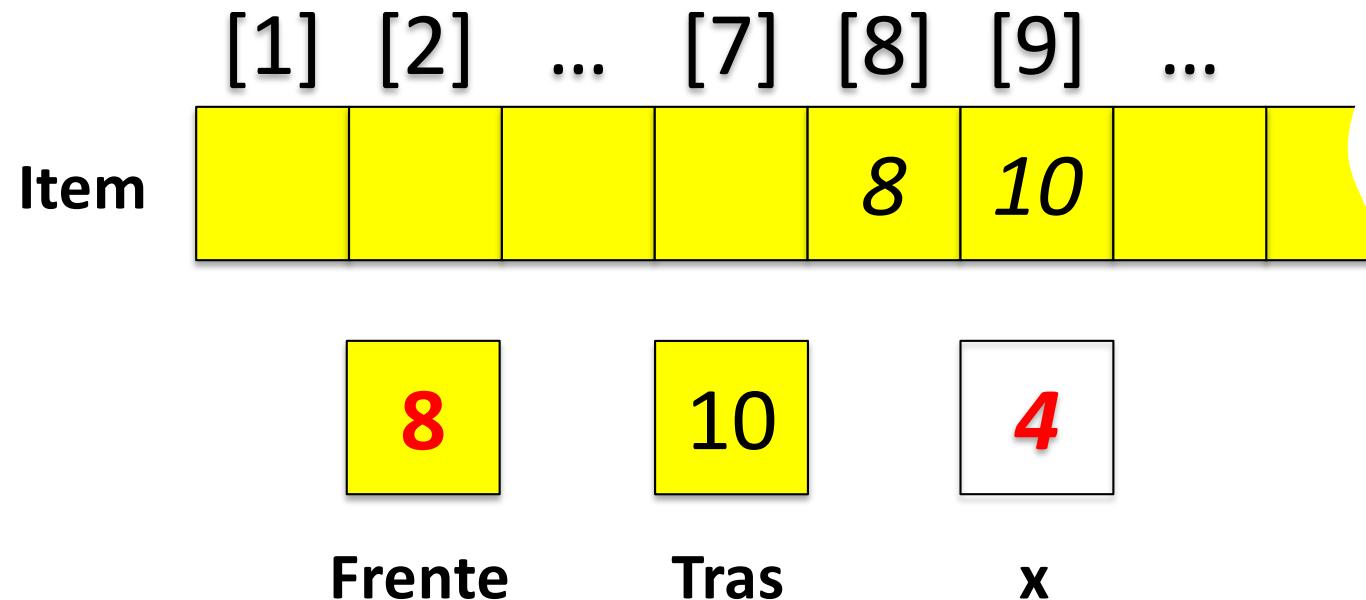


A fila está vazia?

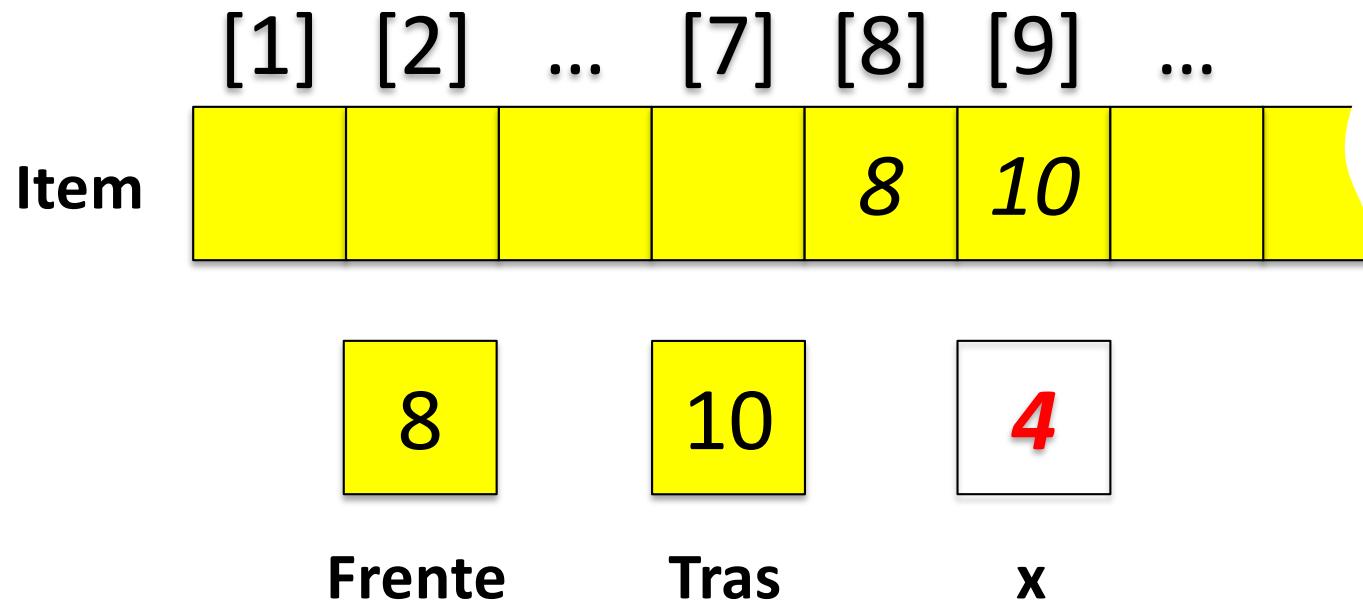
# Retirada de Elementos na Fila



# Retirada de Elementos na Fila



# Retirada de Elementos na Fila



# Operações na Fila por Arranjos

```
void TFila_Inicia(TFila *pFila)
{
    pFila->Frente = 0;
    pFila->Tras = 0;
} /* TFila_Inicia */

int TFila_EhVazia(TFila *pFila)
{
    return (pFila->Frente == pFila->Tras);
} /* TFila_EhVazia */

int TFila_Tamanho(TFila *pFila)
{
    return ((pFila->Tras >= pFila->Frente) ?
            (pFila->Tras - pFila->Frente) :
            (MAXTAM + pFila->Tras - pFila->Frente));
} /* TFila_Tamanho */
```

# Operações na Fila por Arranjos

```
int Tfila_Enfileira(Tfila *pFila, int x) {
    if (pFila->Frente == (pFila->Tras + 1) % MAXTAM)
        return 0; /* fila cheia */

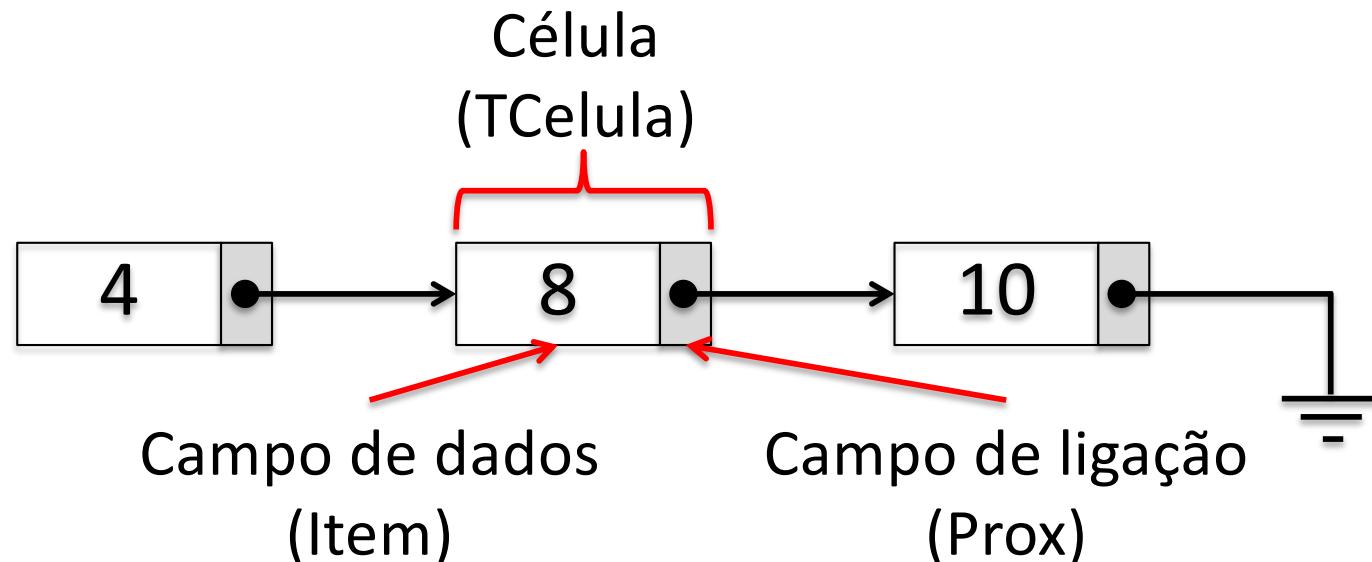
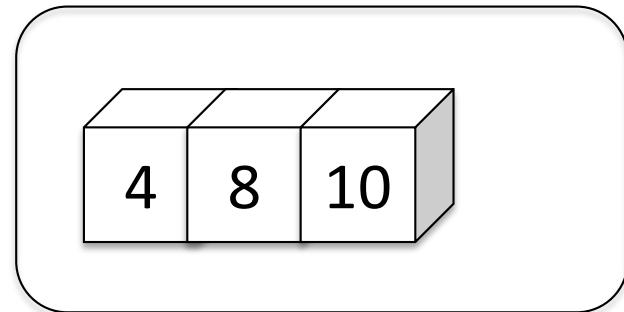
    pFila->Item[pFila->Tras] = x;
    pFila->Tras = (pFila->Tras + 1) % MAXTAM;

    return 1;
}

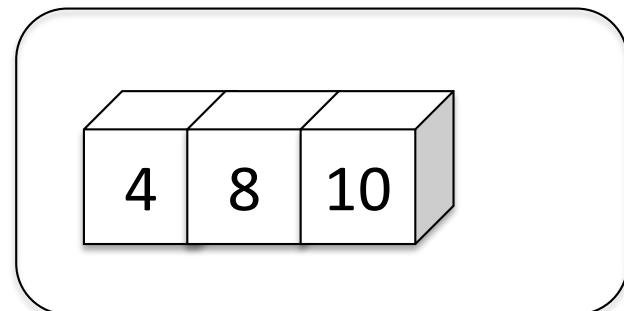
int Tfila_Desenfileira(Tfila *pFila, int *pX) {
    if (Tfila_EhVazia(pFila))
        return 0;
    *pX = pFila->Item[pFila->Frente];
    pFila->Frente = (pFila->Frente + 1) % MAXTAM;
    return 1;
}
```

# IMPLEMENTAÇÃO POR APONTADORES

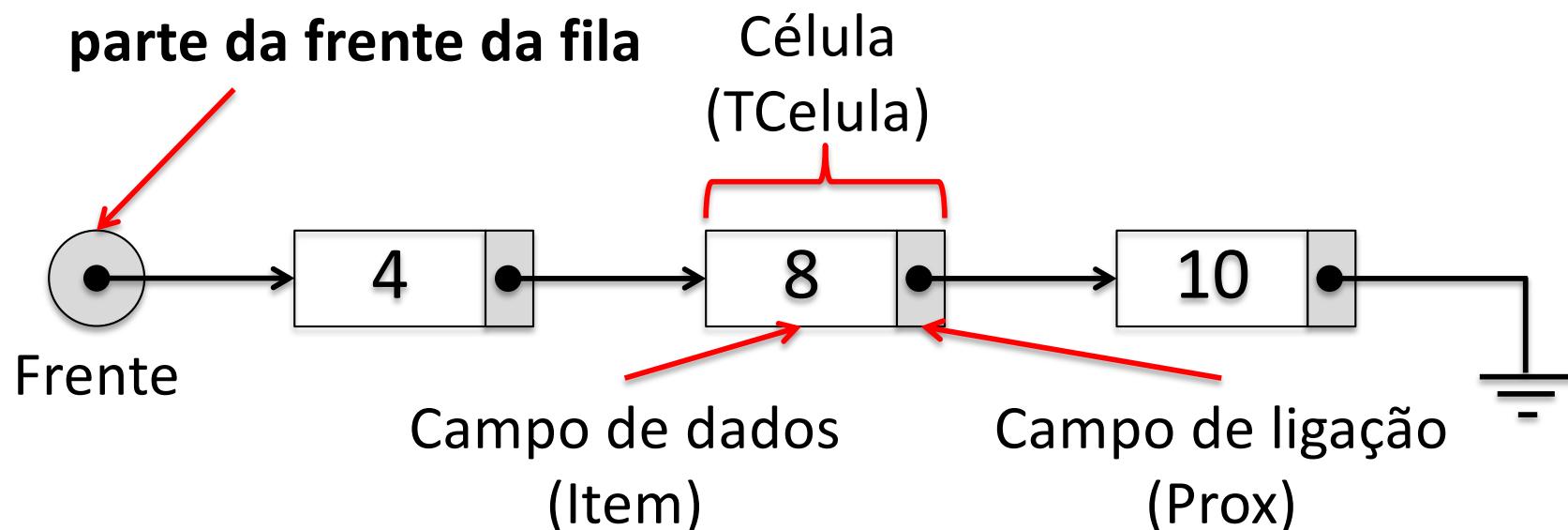
- Os itens são colocados em uma estrutura (**TCelula**) que contém um campo com um item (**Item**) e outro campo com um apontador para a próxima célula (**Prox**)



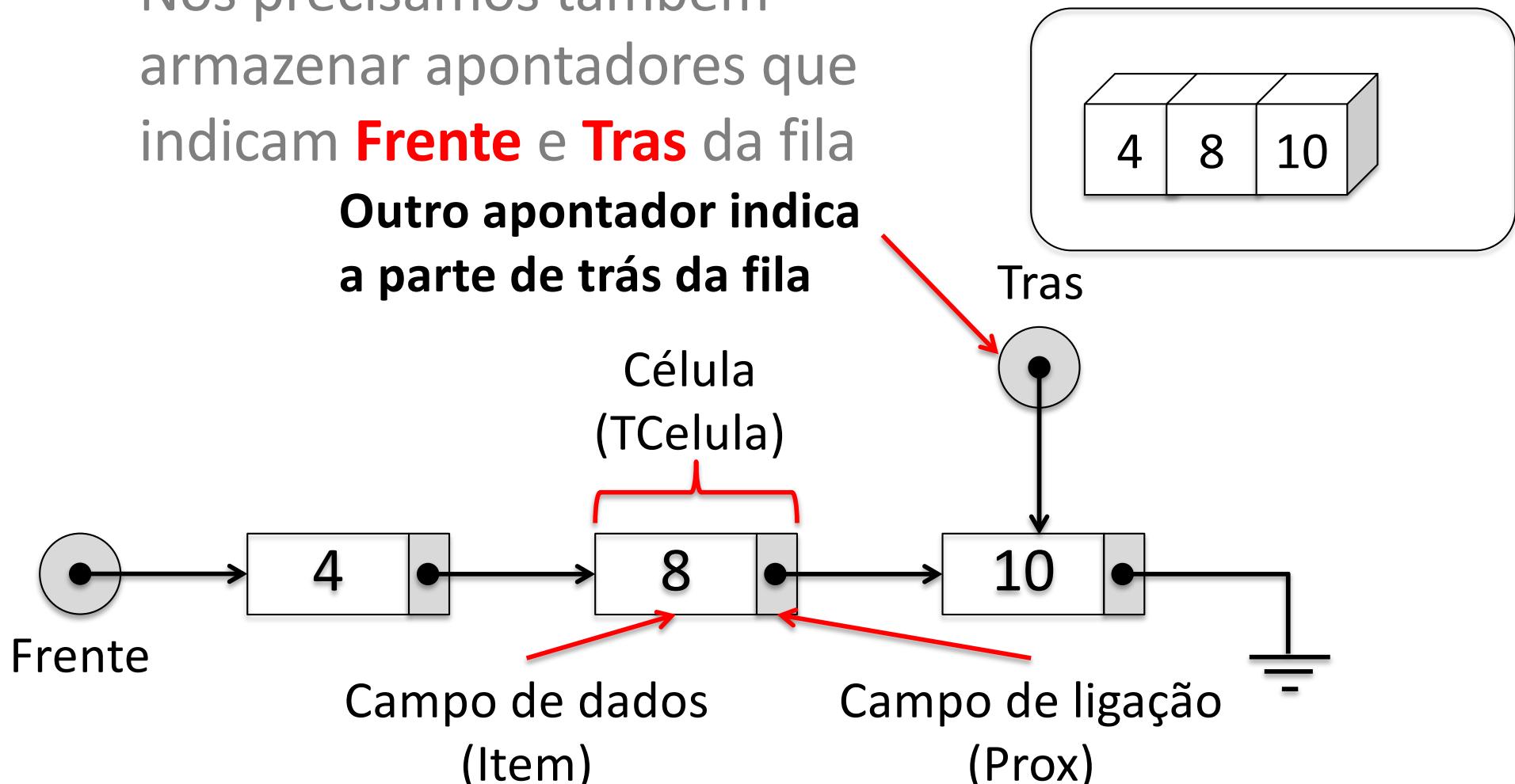
- Nós precisamos também armazenar apontadores que indicam **Frente** ...



**Um apontador indica a parte da frente da fila**

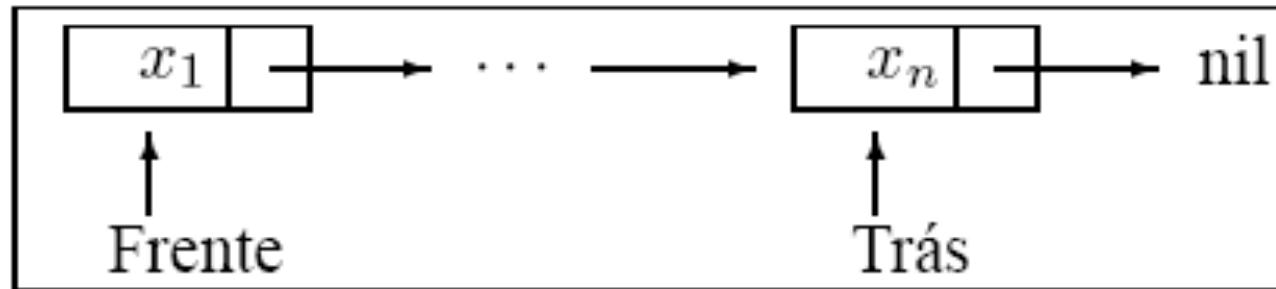


- Nós precisamos também armazenar apontadores que indicam **Frente** e **Tras** da fila  
**Outro apontador indica a parte de trás da fila**



# Estrutura da Fila por Apontador

- Cada item é encadeado com o seguinte por meio de um apontador
- Permite utilizar posições não contíguas de memória



- O campo **Tamanho** evita a contagem do número de itens na função **TFila\_Tamanho**
- Cada célula contém um item da fila e um apontador para outra célula
- O registro **TFila** contém um apontador para a **Frente** da fila e um apontador para a parte de **Tras** da fila

# Estrutura da Fila por Apontador

```
typedef int TChave;

typedef struct {
    TChave Chave;
    /* outros componentes */
} TItem;

typedef struct SCelula *TApontador;

typedef struct SCelula {
    TItem Item;
    TApontador Prox;
} TCelula;

typedef struct {
    TApontador Frente, Tras;
    int Tamanho;
} TFila;
```

Tamanho

0

Frente

NULL

NULL

Tras

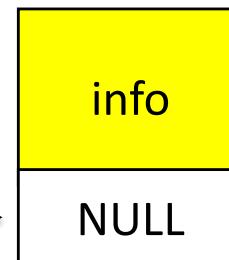
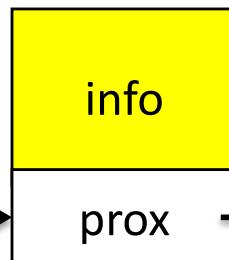
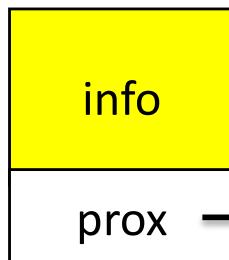
# Inserção de Elementos na Fila

Tamanho

3

Frente

—



Tras

—

- Opção única de posição onde pode inserir:
  - Trás da fila, ou seja, última posição

# Inserção de Elementos na Fila

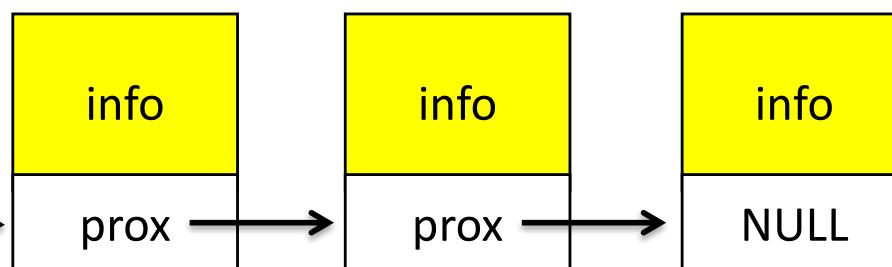
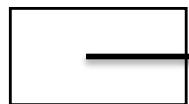
Tamanho

3

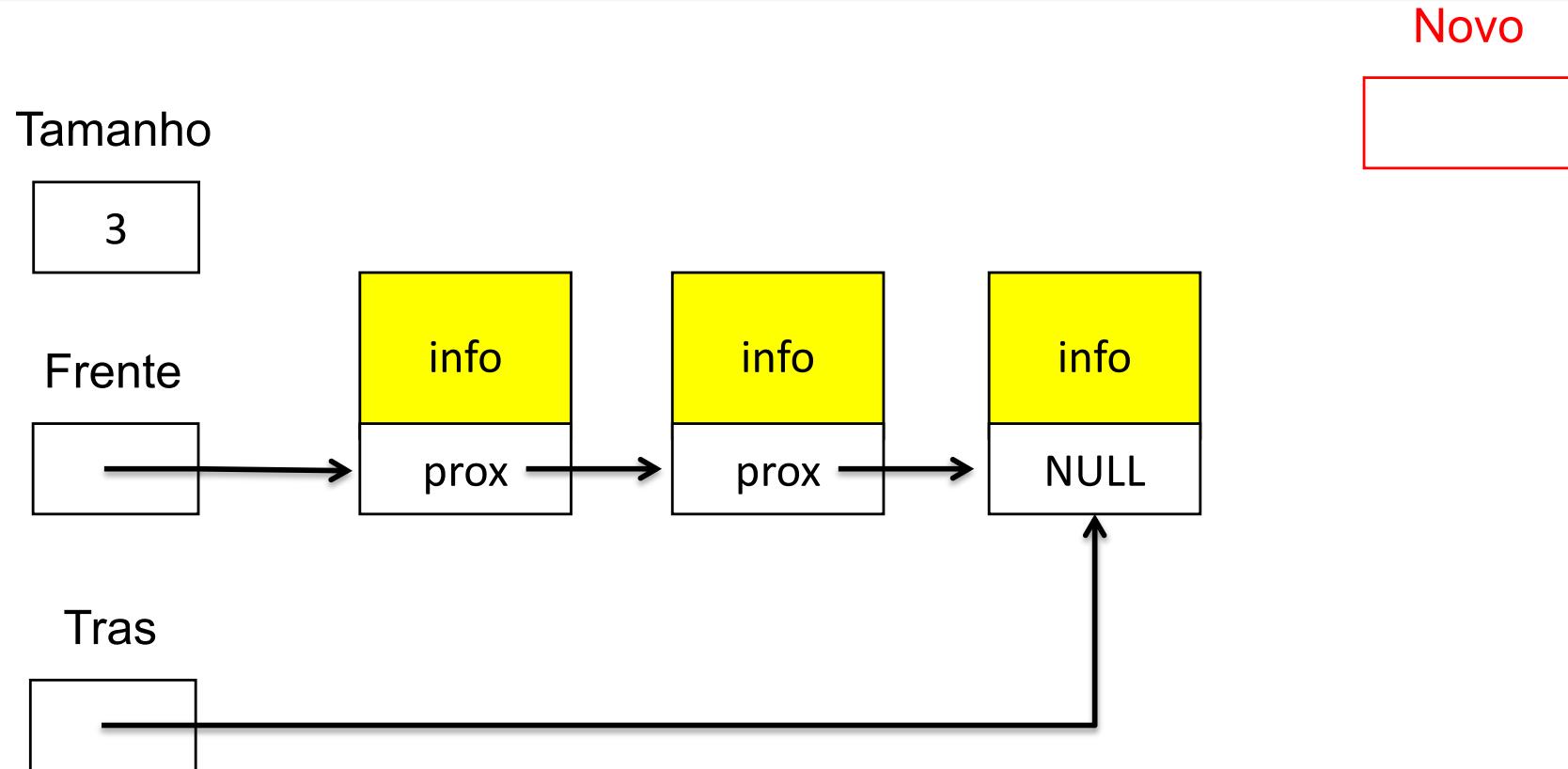
Frente



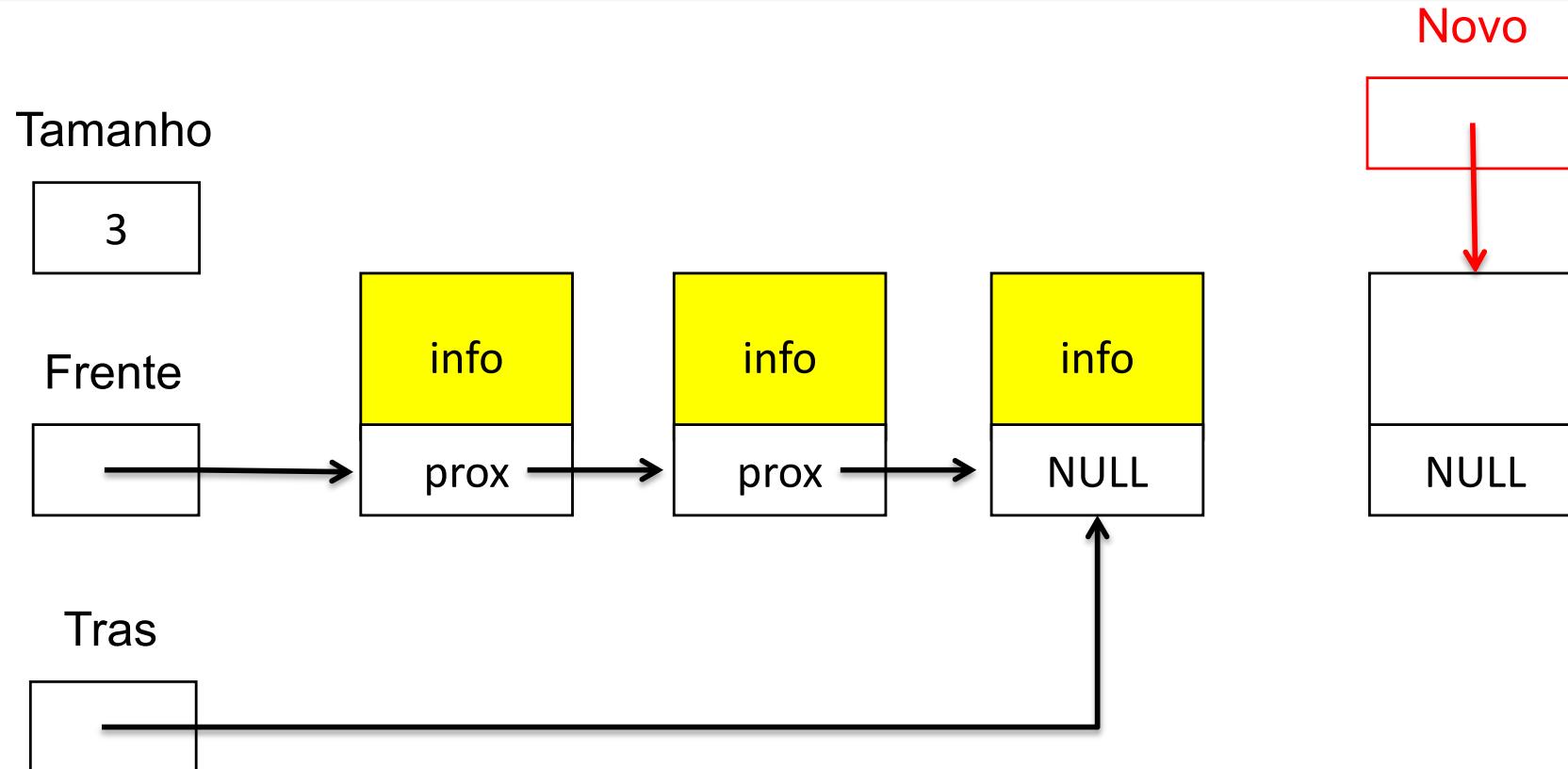
Tras



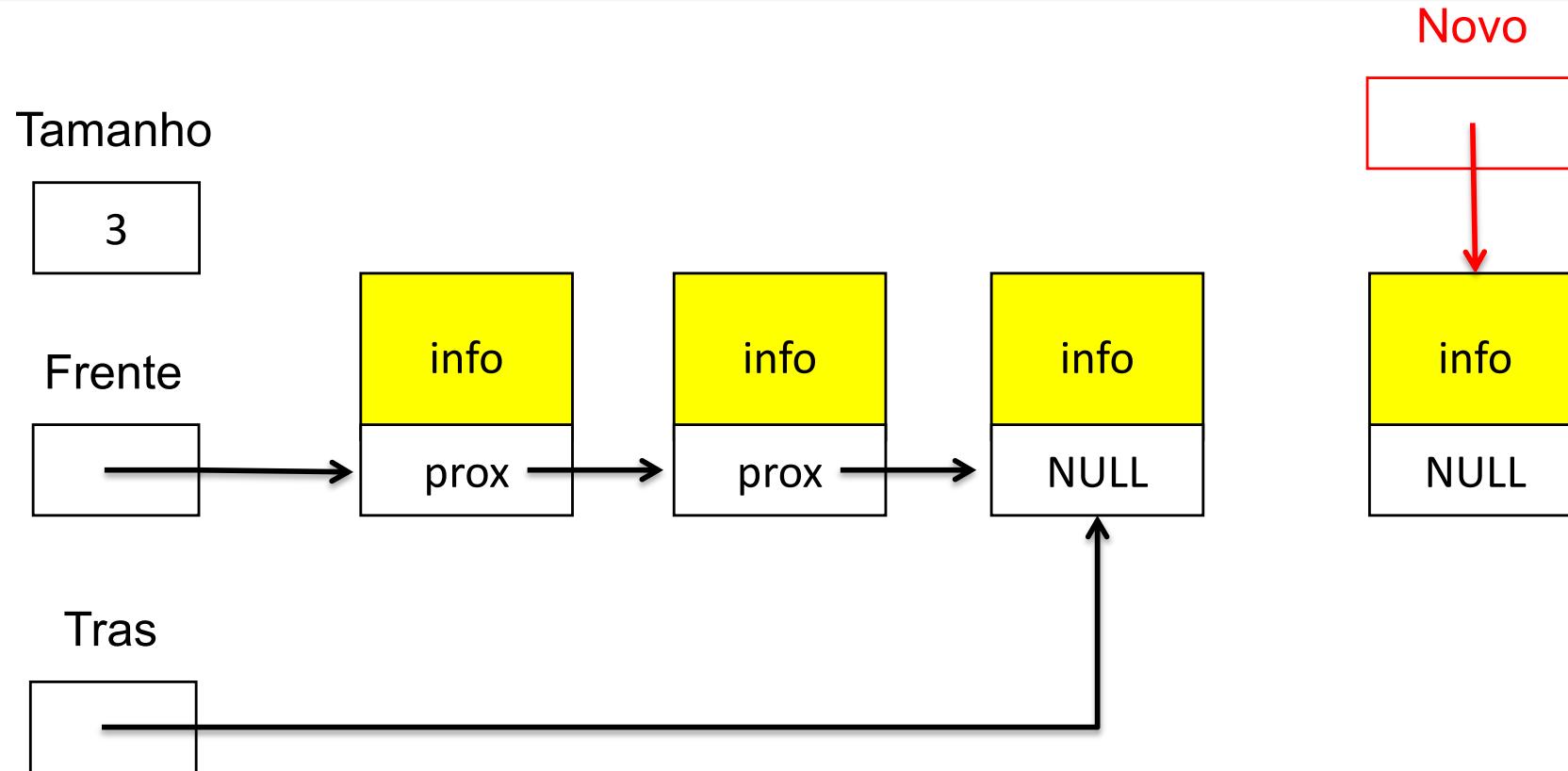
# Inserção de Elementos na Fila



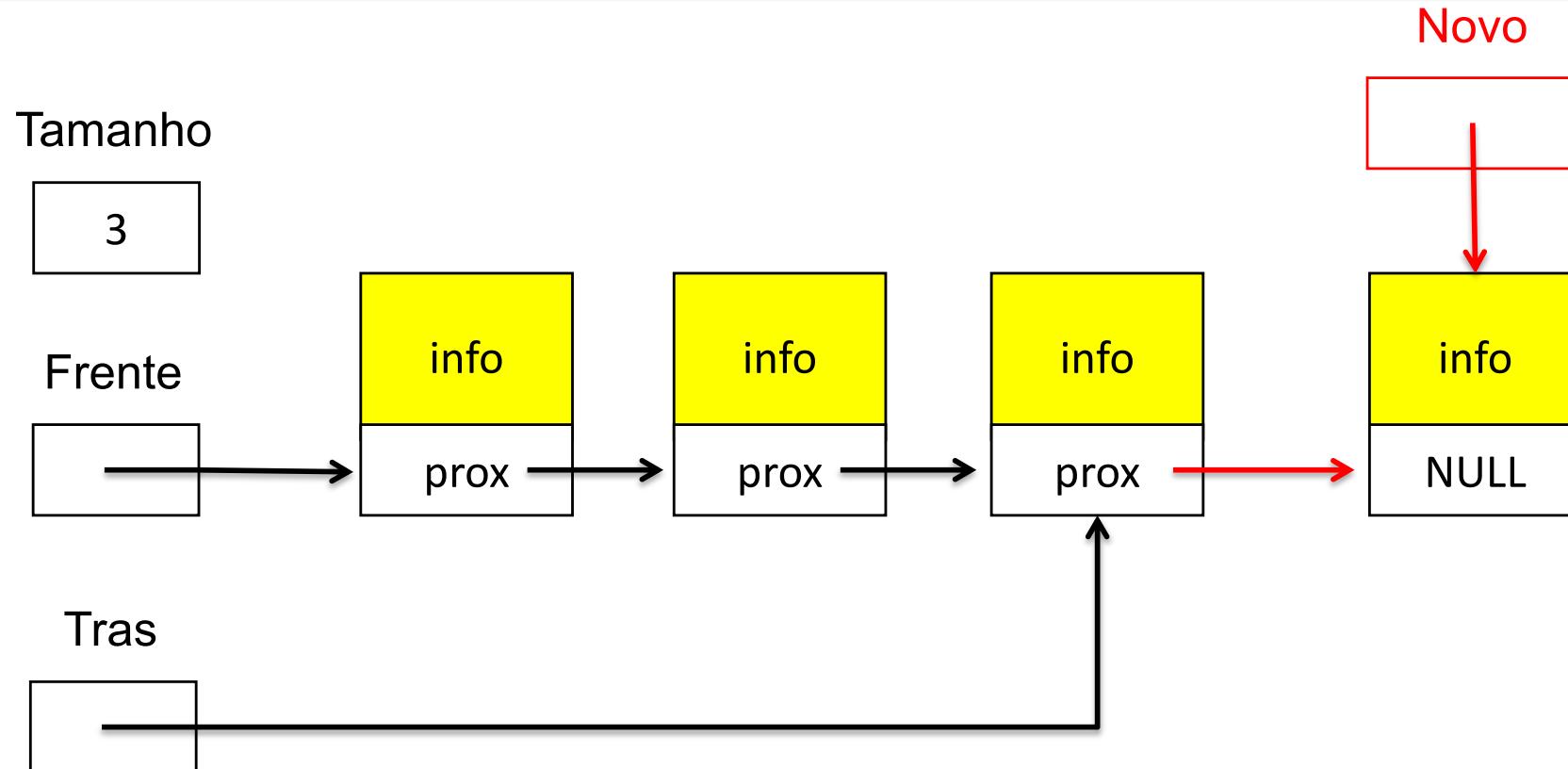
# Inserção de Elementos na Fila



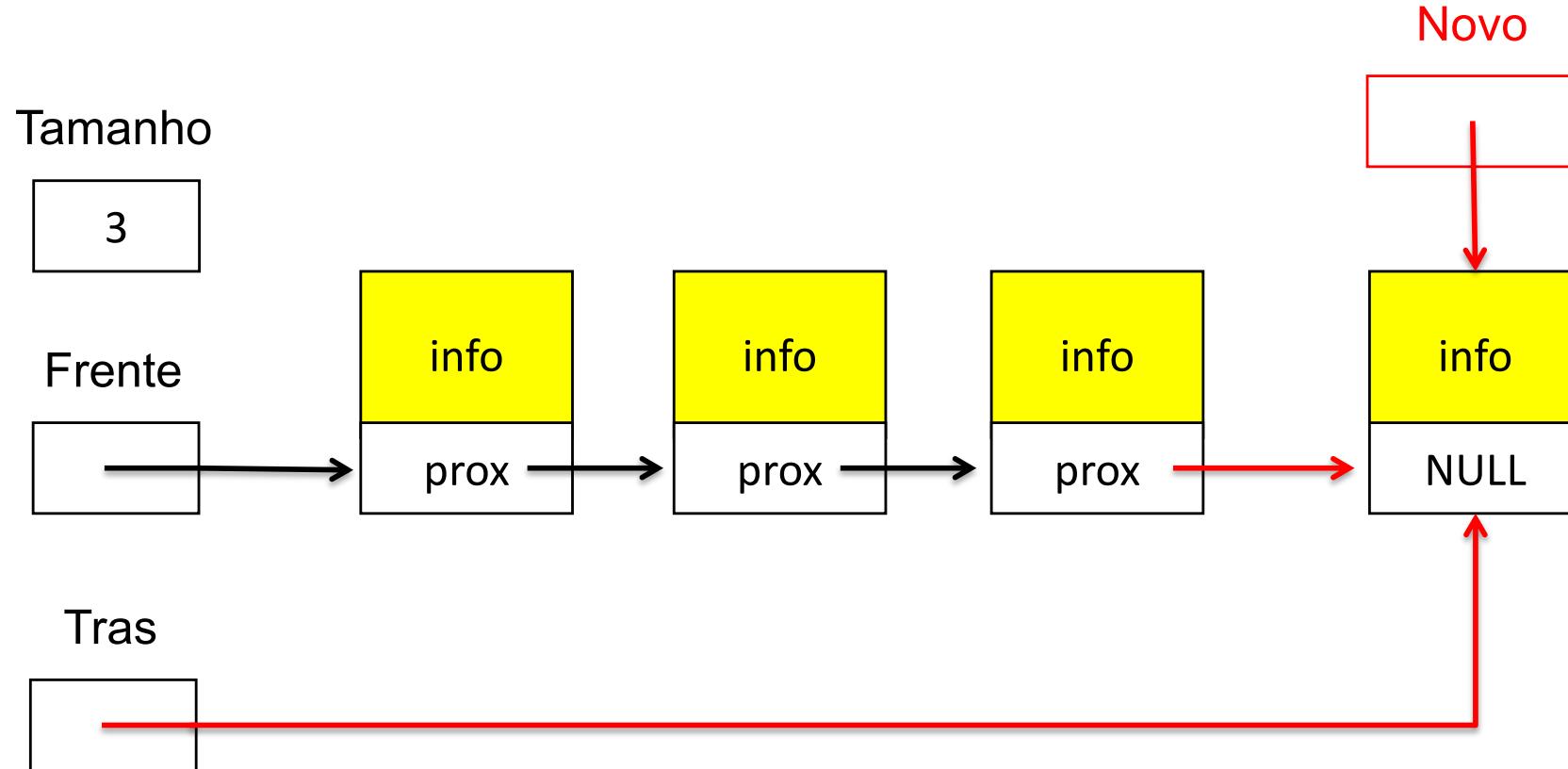
# Inserção de Elementos na Fila



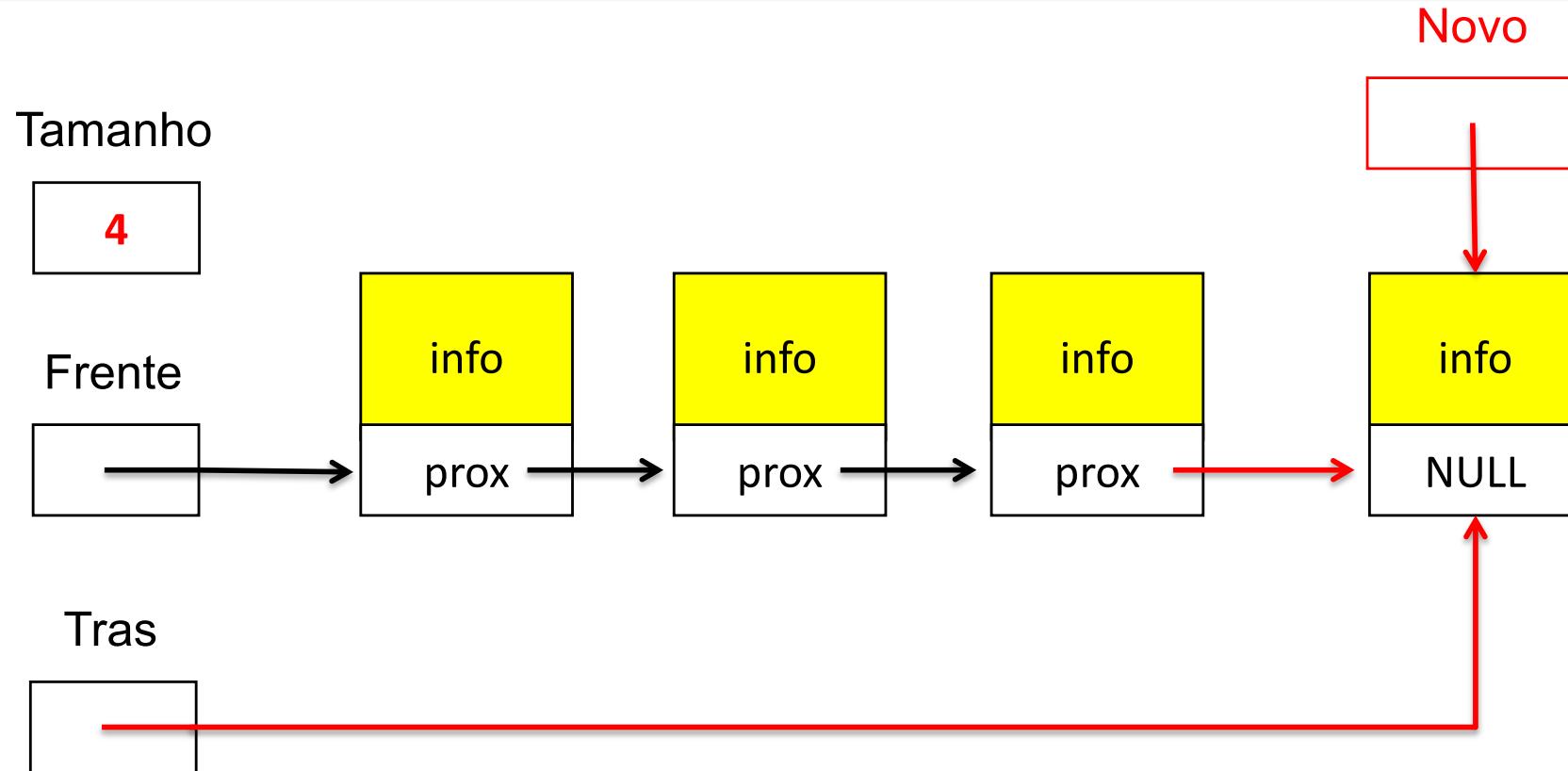
# Inserção de Elementos na Fila



# Inserção de Elementos na Fila



# Inserção de Elementos na Fila



# Inserção de Elementos na Fila

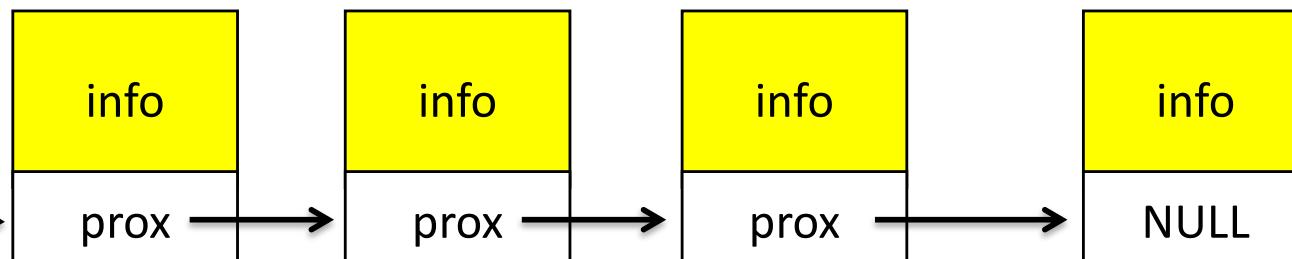
Tamanho

4

Frente



Tras



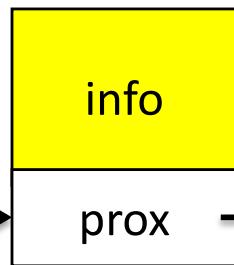
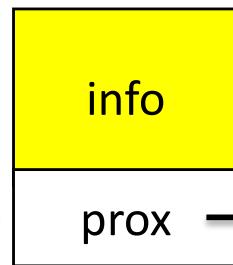
# Retirada de Elementos da Fila

Tamanho

3

Frente

—



Tras

—

- Opção única de posição onde pode retirar:
  - Frente da fila, ou seja, 1<sup>a</sup> posição

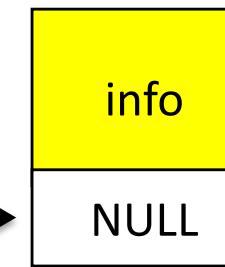
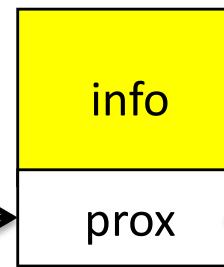
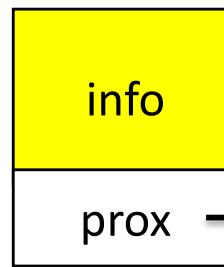
# Retirada de Elementos da Fila

Tamanho

3

Frente

—→



Tras

—→

X

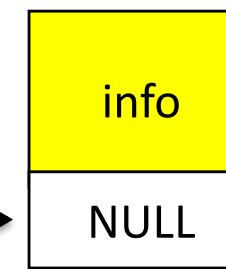
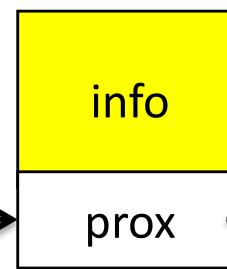
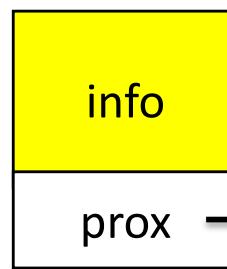


# Retirada de Elementos da Fila

Tamanho

3

Frente



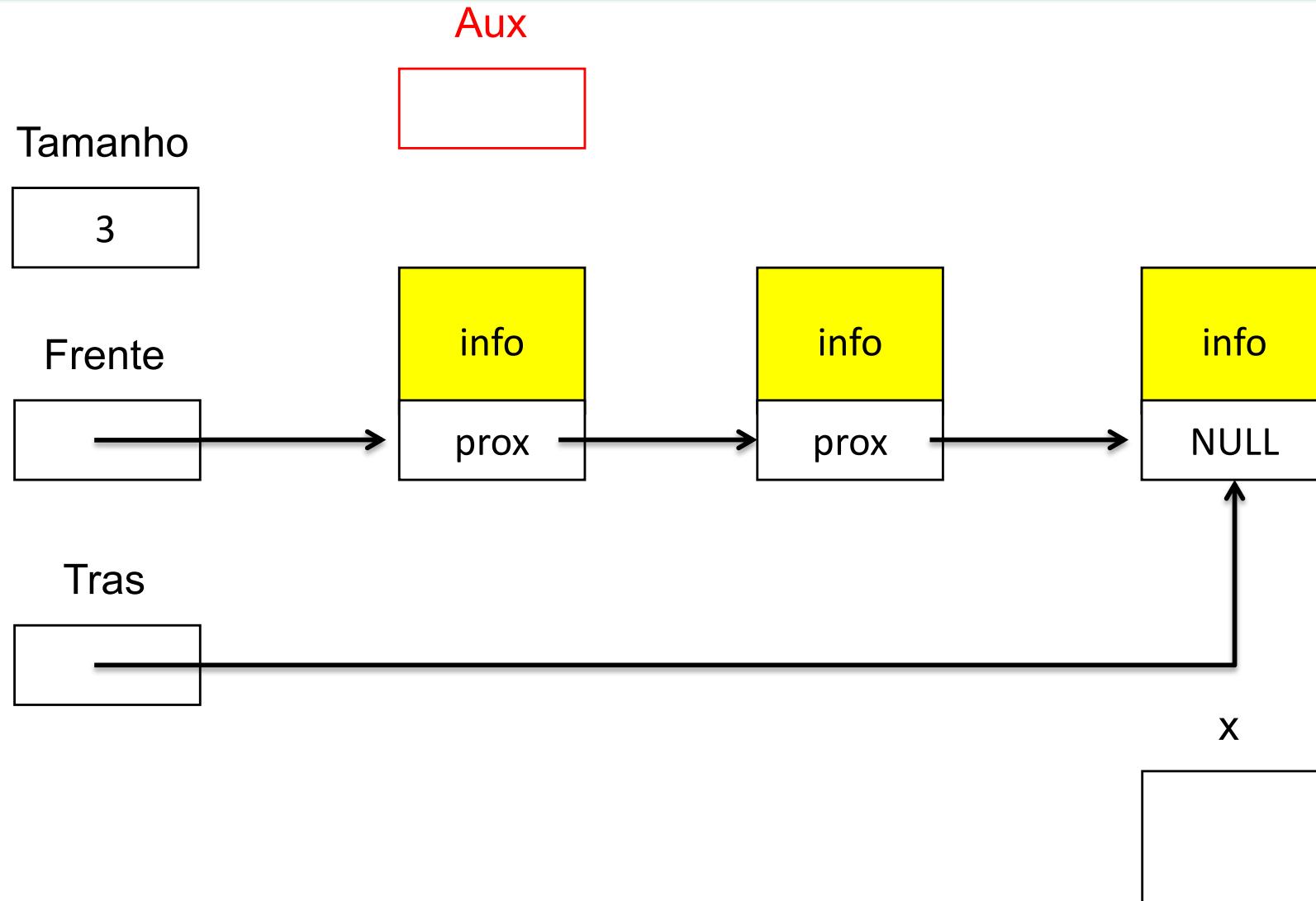
Tras



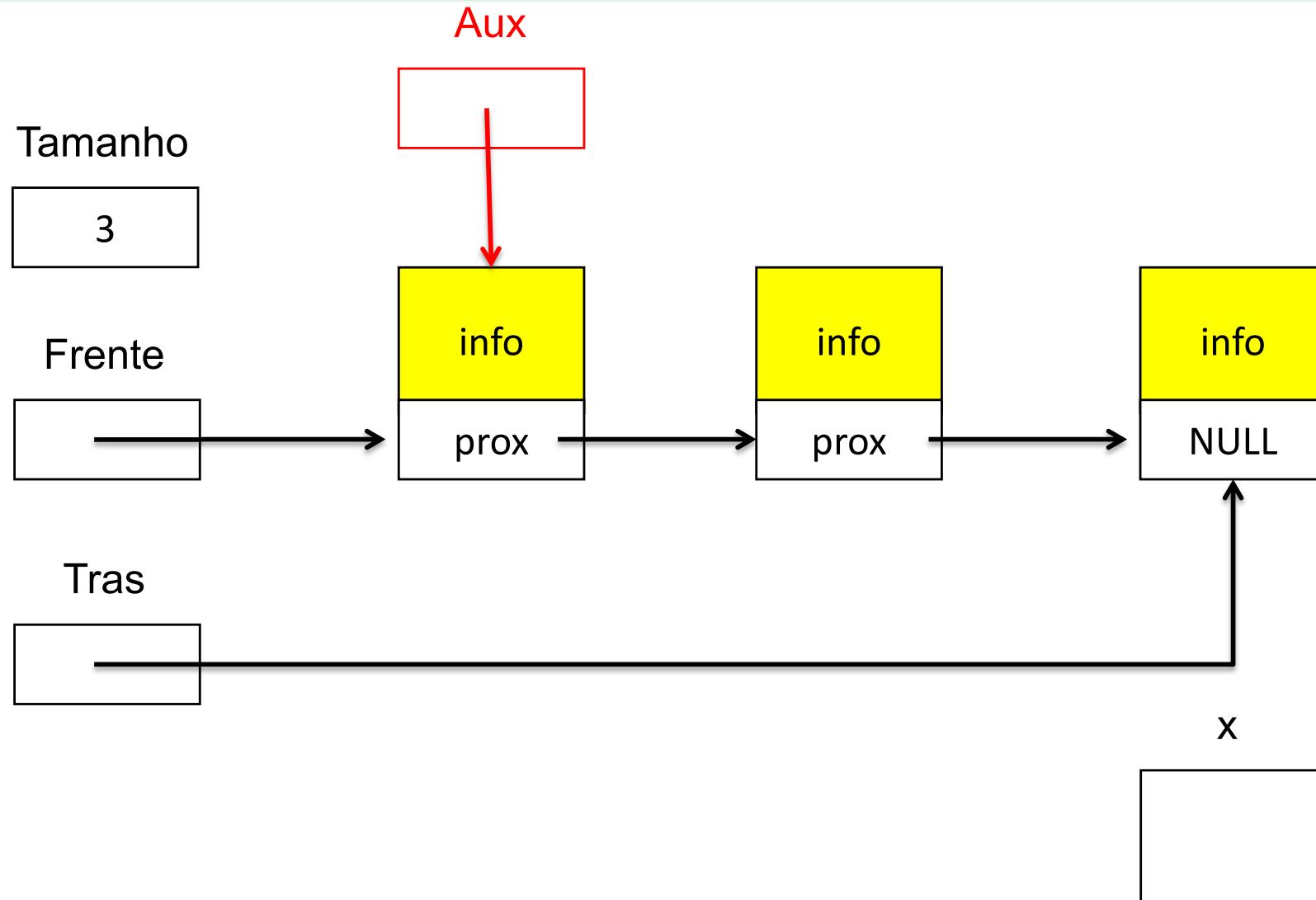
**A fila está vazia?**



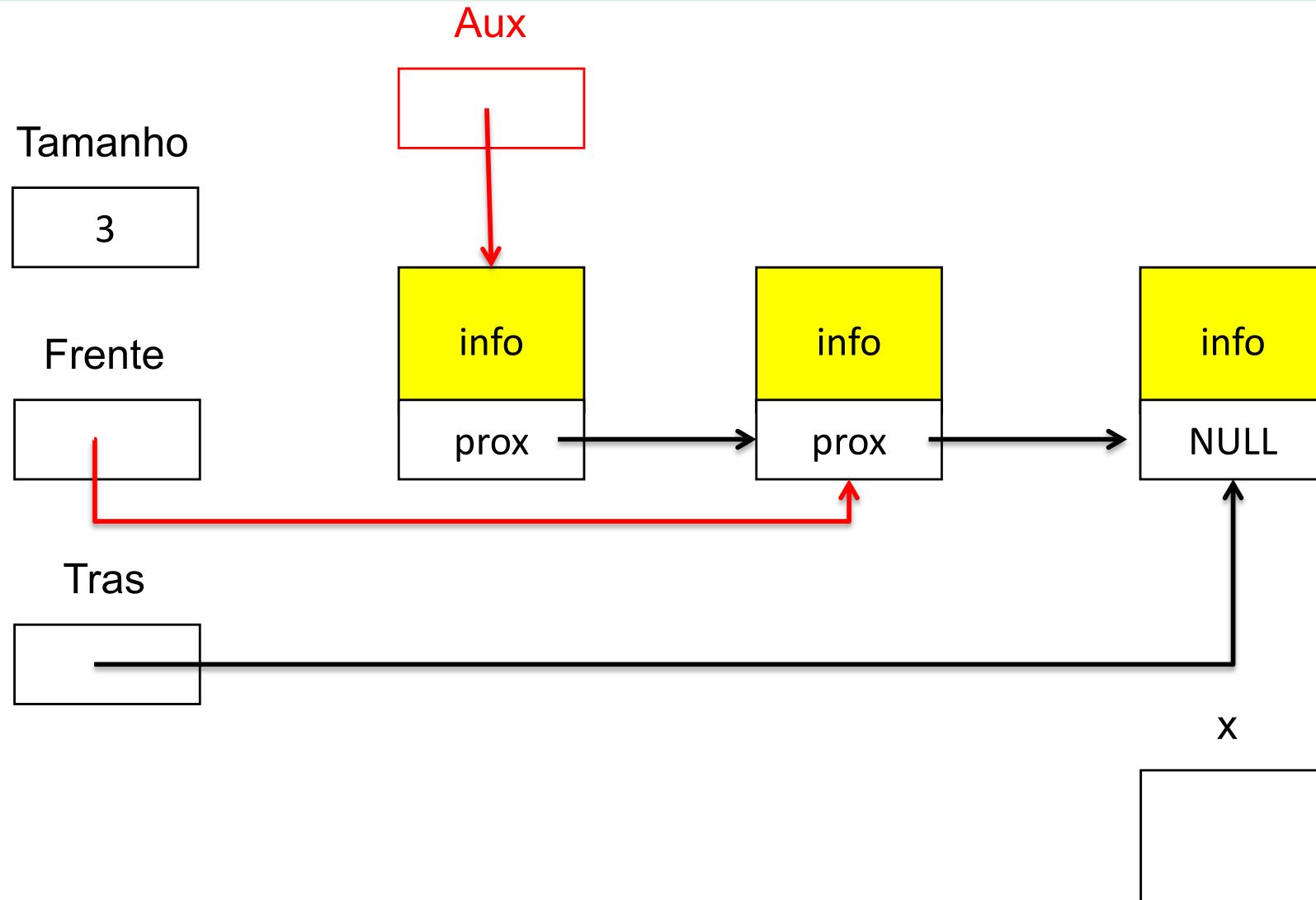
# Retirada de Elementos da Fila



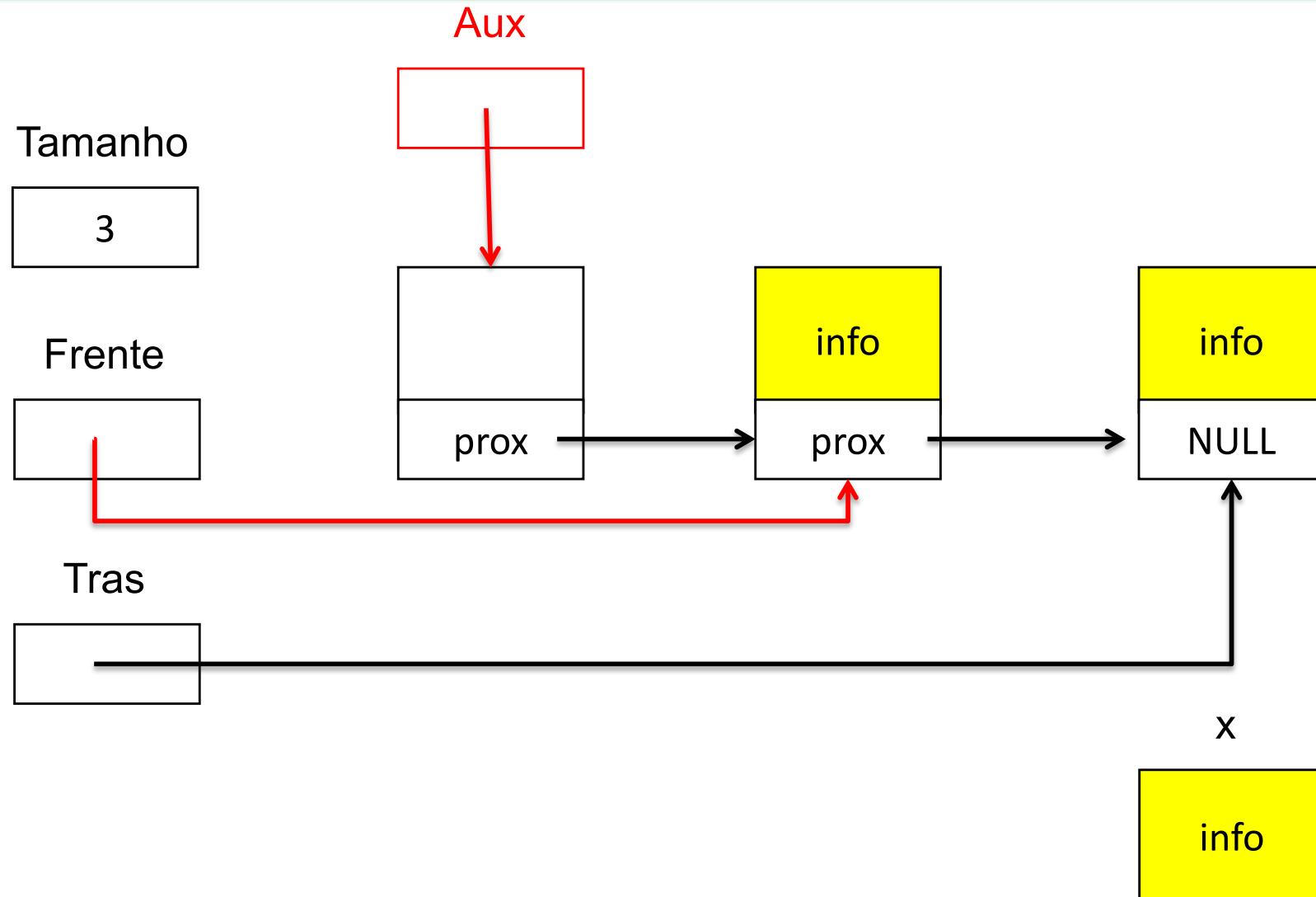
# Retirada de Elementos da Fila



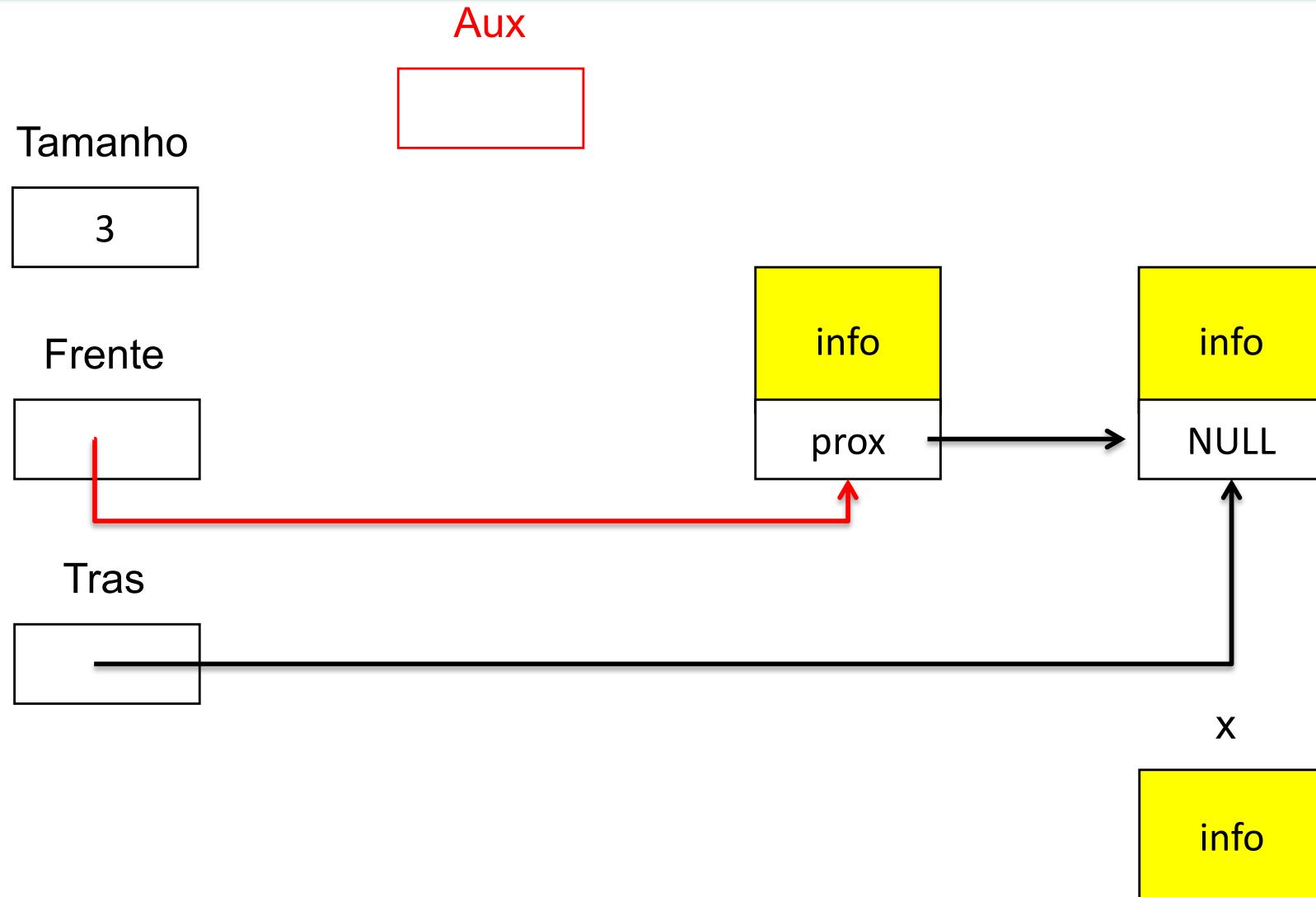
# Retirada de Elementos da Fila



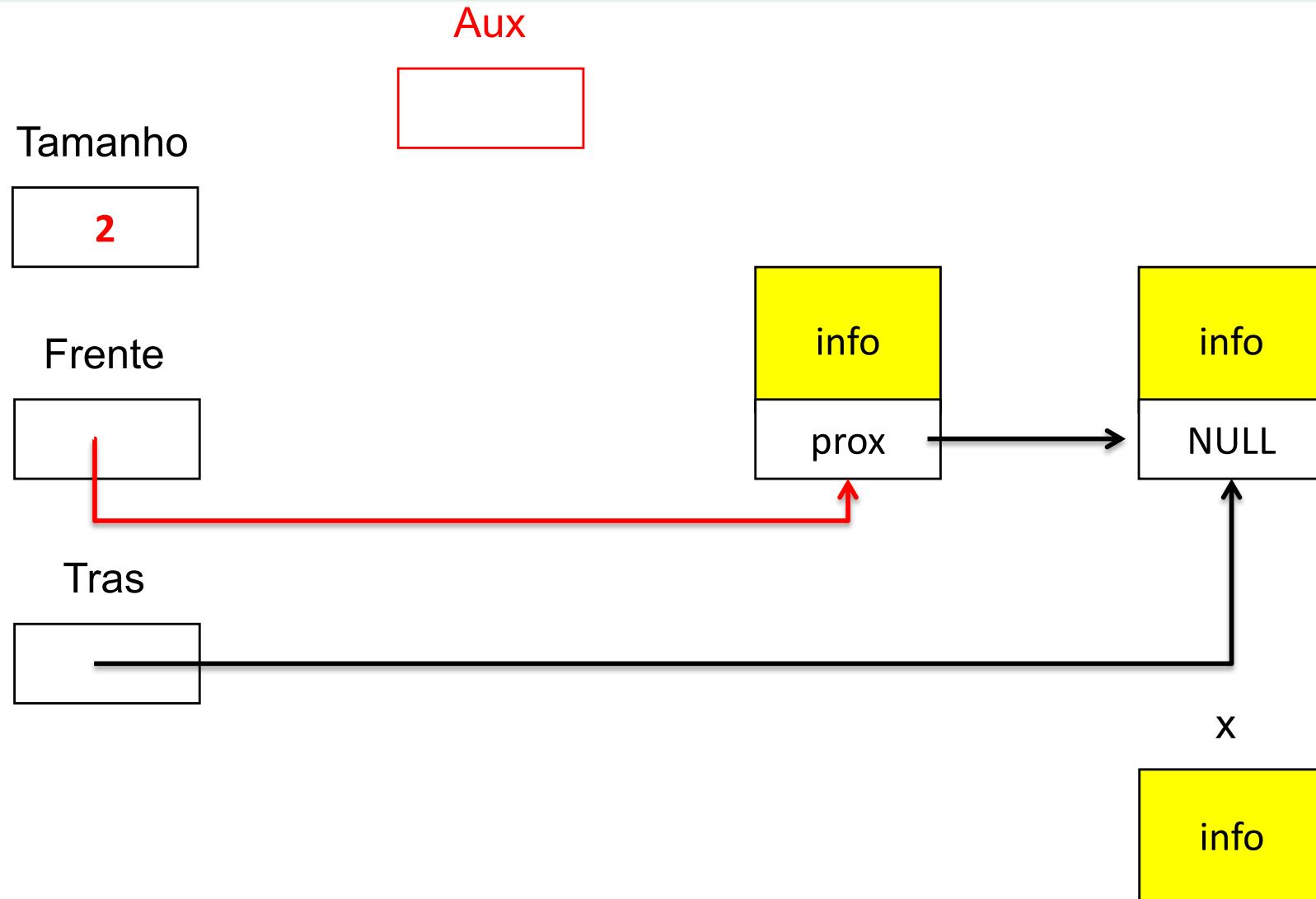
# Retirada de Elementos da Fila



# Retirada de Elementos da Fila



# Retirada de Elementos da Fila

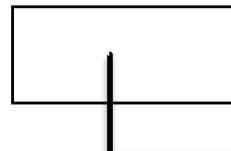


# Retirada de Elementos da Fila

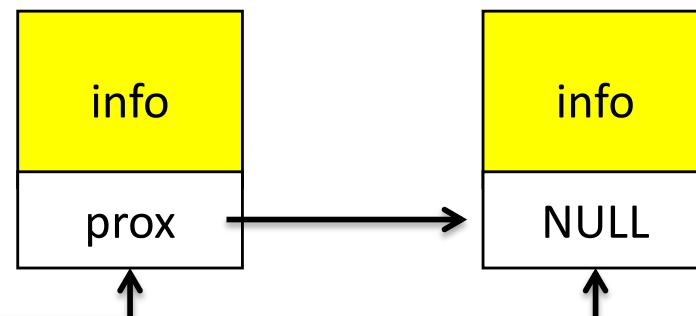
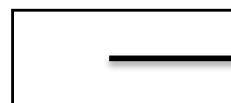
Tamanho

2

Frente



Tras



X

info

# Operações em Filas

```
/* Inicia as variaveis da fila */
void TFila_Inicia(TFila *pFila)
{
    pFila->Frete = NULL;
    pFila->Tras = pFila->Frete;
    pFila->Tamanho = 0;
}

/* Retorna se a fila esta vazia */
int TFila_EhVazia(TFila *pFila)
{
    return (pFila->Frete == NULL);
}

/* Retorna o tamanho da fila */
int TFila_Tamanho(TFila *pFila)
{
    return (pFila->Tamanho);
```

# Operações em Filas

```
/* Enfileira um item na parte de tras da fila */
int TFila_Enfileira(TFila *pFila, TItem x)
{
    TApontador pNovo;

    pNovo = (TApontador) malloc(sizeof(TCelula));
    pNovo->Item = x;
    pNovo->Prox = NULL;

    if (TFila_EhVazia(pFila))
        pFila->Frente = pNovo;
    else
        pFila->Tras->Prox = pNovo;
    pFila->Tras = pNovo;
    pFila->Tamanho++;
    return 1;
}
```

# Operações em Filas

```
/* Desenfileira o item da frente da fila */
int TFila_Desenfileira(TFila *pFila, TItem *pX)
{
    TA pontador pAux;

    if (TFila_EhVazia(pFila))
        return 0;

    pAux = pFila->Frente;
    pFila->Frente = pAux->Prox;
    *pX = pAux->Item;
    free(pAux);
    pFila->Tamanho--;

    return 1;
}
```

- Imagine que você descobriu uma biblioteca que implementa o TAD Pilha de maneira diferente, mas ainda oferece suporte às seguintes operações:

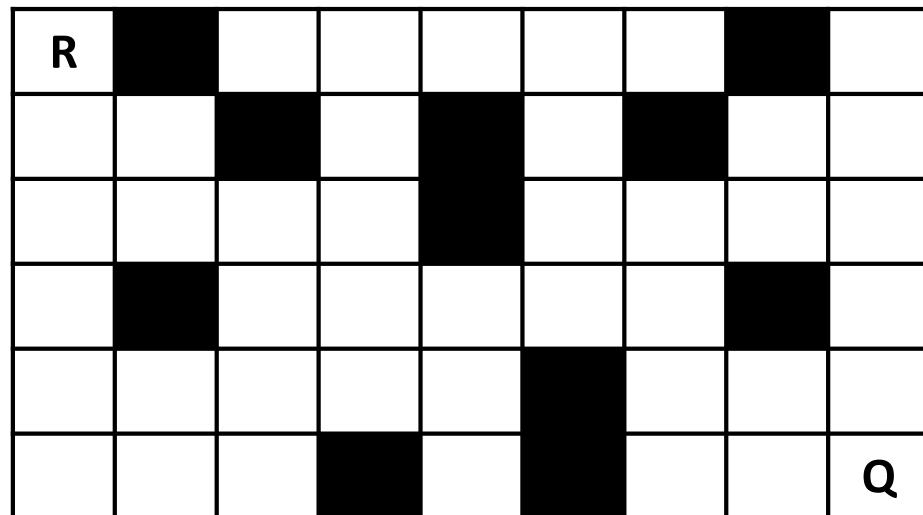
```
void TPilha_Inicia(TPilha *pPilha);
int TPilha_EhVazia(TPilha *pPilha);
int TPilha_Empilha(TPilha *pPilha, TItem x);
int TPilha_Desempilha(TPilha *pPilha, TItem *pX);
int TPilha_Tamanho(TPilha *pPilha);
```

- Você possui apenas os códigos em binário dessa biblioteca e, portanto, não tem conhecimento de como esse TAD Pilha foi implementado
- Usando esse TAD Pilha, implemente as operações do TAD Fila visto em aula usando somente duas pilhas

# Desafio

## ■ Problema do Ratinho

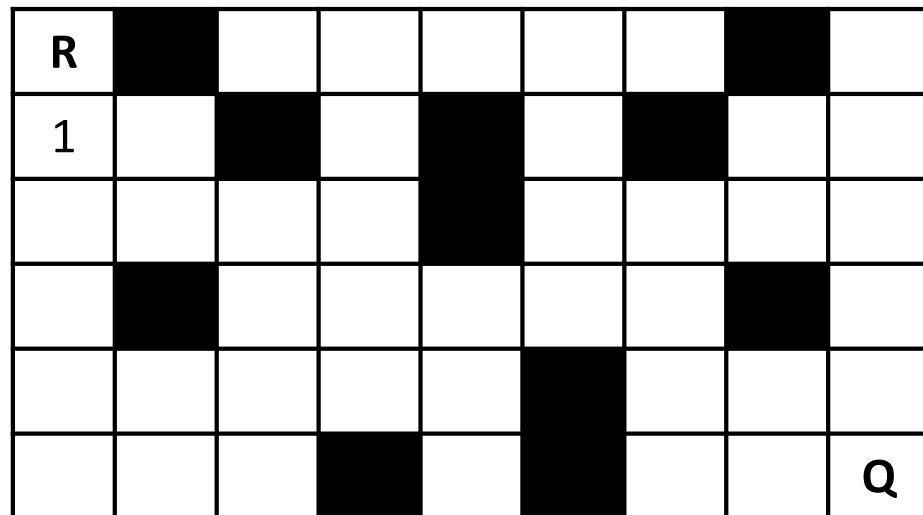
- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.



# Desafio

## ■ Problema do Ratinho

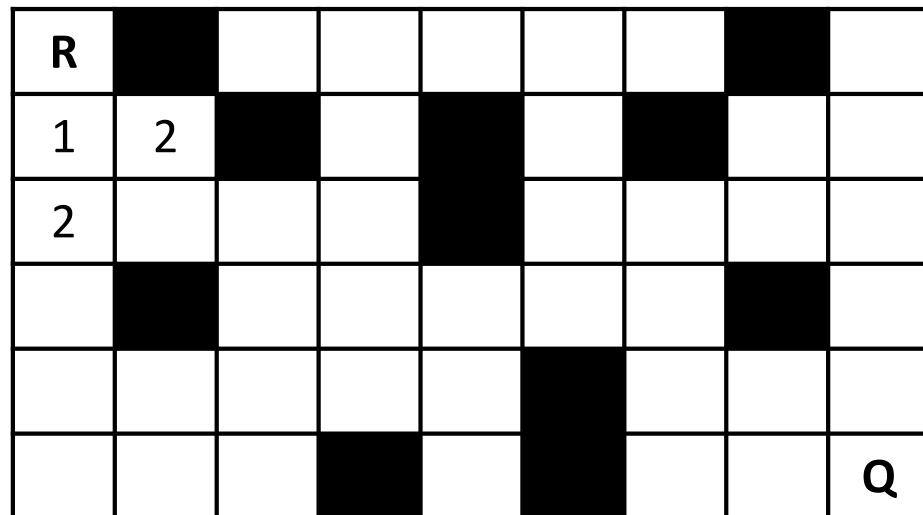
- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.



# Desafio

## ■ Problema do Ratinho

- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.



# Desafio

## ■ Problema do Ratinho

- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.

R								
1	2							
2	3							
3								
								Q

# Desafio

## ■ Problema do Ratinho

- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.

R								
1	2							
2	3	4						
3								
4								Q

# Desafio

## ■ Problema do Ratinho

- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.

R								
1	2							
2	3	4	5					
3			5					
4	5							
5								Q

# Desafio

## ■ Problema do Ratinho

- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.

R		8	7	8	9	10		14
1	2		6		10		12	13
2	3	4	5		9	10	11	12
3		5	6	7	8	9		13
4	5	6	7	8		10	11	12
5	6	7		9		11	12	Q

# Desafio

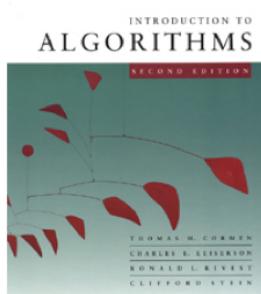
## ■ Problema do Ratinho

- **Problema:** Dado um labirinto e as posições do rato e do queijo no labirinto, determine o número mínimo de passos que o rato precisa para chegar até o queijo.
- **Entrada:** inteiros  $m$  e  $n$ , matriz  $A_{m \times n}$  tal que

$$A[i, j] = \begin{cases} -1 & \text{se } (i, j) \text{ é parede} \\ 0 & \text{se } (i, j) \text{ é livre} \end{cases}$$

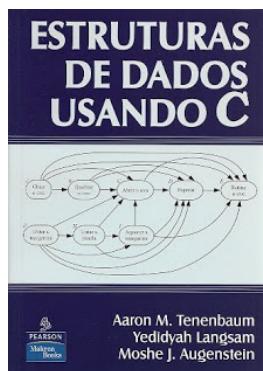
e posições  $(x_r, y_r)$  do rato e  $(x_q, y_q)$  do queijo

- **Saída:** a distância do rato até o queijo



CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms.** 3<sup>a</sup> Edição. MIT Press, 2009. **Seção 10.1**

ZIVIANI, N. **Projeto de Algoritmos com Implementações em Pascal e C.** 3<sup>a</sup> Edição. Cengage Learning, 2010. **Seção 3.3**



TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. **Estruturas de Dados usando C.** Pearson Makron Books, 2008.

**Capítulo 4**