INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CATARINENSE
Campus Videira

Filas

Professor: Manassés Ribeiro

manasses.ribeiro@ifc.edu.br

Agenda

- Conceitos
- Aspectos de Implementação usando Vetores
- Aspectos de Implementação usando Lista Encadeada
- Fila de Prioridades

Conceitos

Conceito

As **filas** são **estruturas de dados** cujos elementos estão **ordenados** com base na **sequência** na qual foram inseridos.

Conceito

- Conjunto ordenado de itens onde:
  - **insere-se** itens **por uma** extremidade (chamada final da fila);
  - **remove-se** itens **por outra** extremidade (chamada início da fila);
- Ou seja:
  - o primeiro elemento inserido é o primeiro a ser removido.
    - lista **fifo** *(**f**irst-**i**n, **f**irst-**o**ut)*

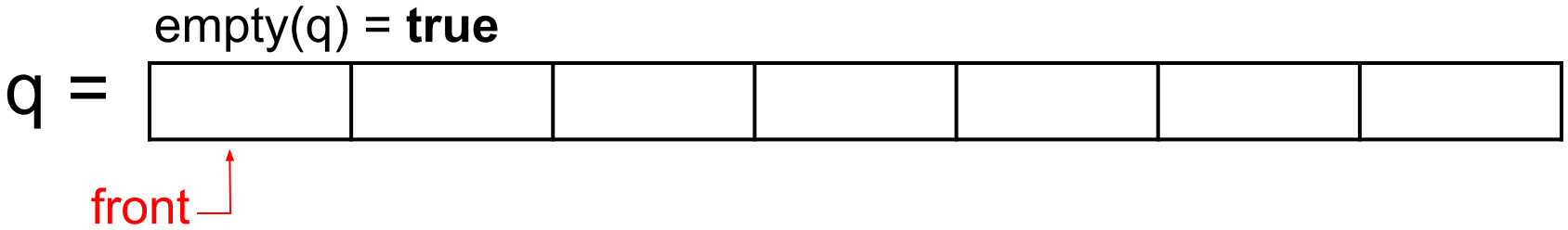
Conceito

- Exemplos:
  - fila de banco;
  - fila em ponto de ônibus;
  - um grupo de carros aguardando sua vez no pedágio.

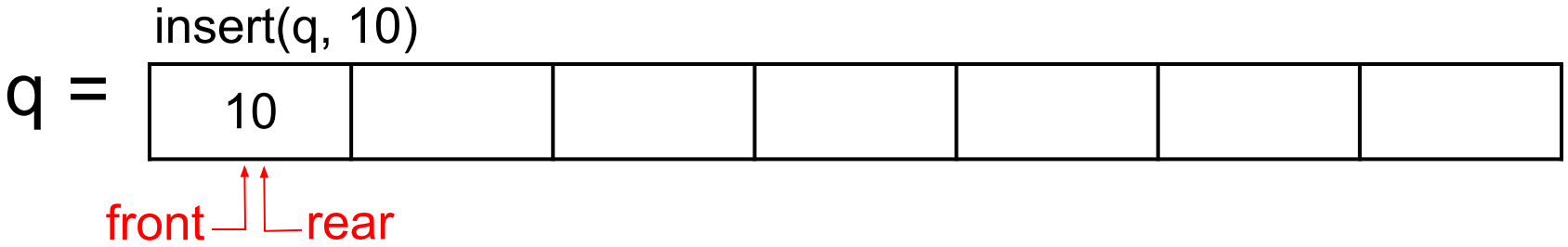
Operações

- São três as operações primitivas: **insert**, **remove** e **empty**;
  - **insert(q, x):** insere o item **x** no fim da fila **q**.
  - **remove(q):** remove o primeiro elemento da fila **q** e retorna o conteúdo removido;
  - **empty(q):** indica se a fila está vazia e retorna verdadeiro ou falso dependendo de a fila conter ou não algum elemento.

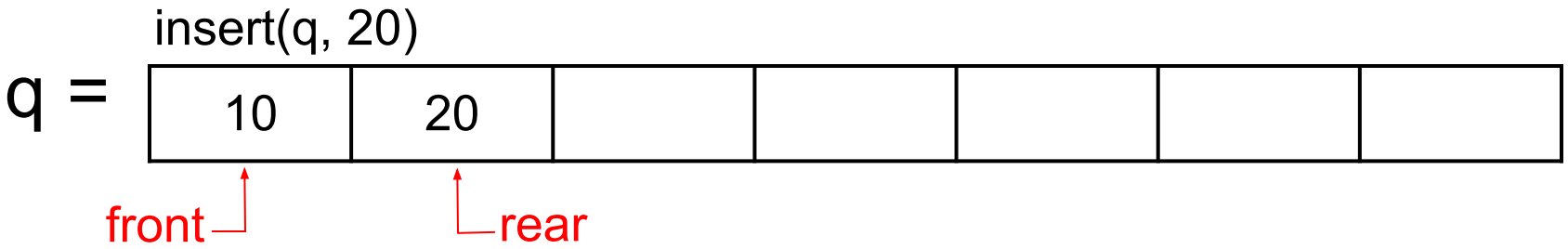
Exemplo de operações



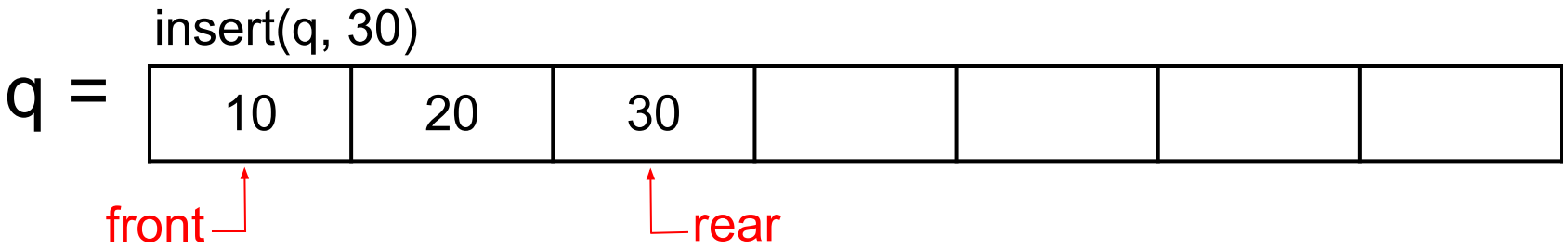
Exemplo de operações



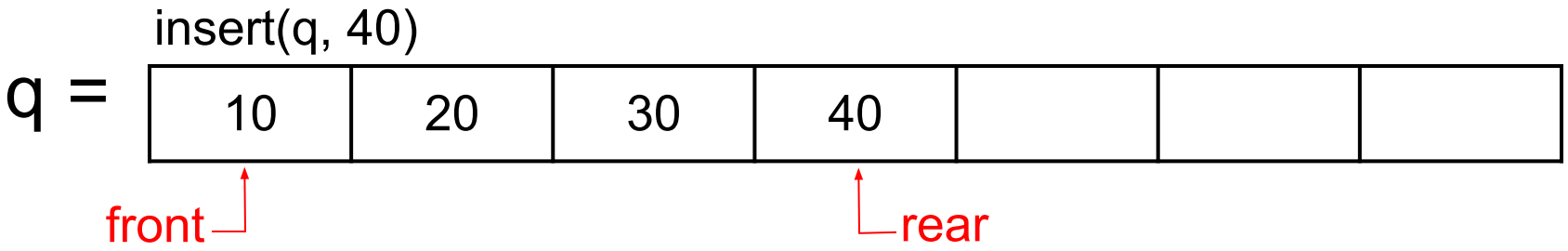
Exemplo de operações



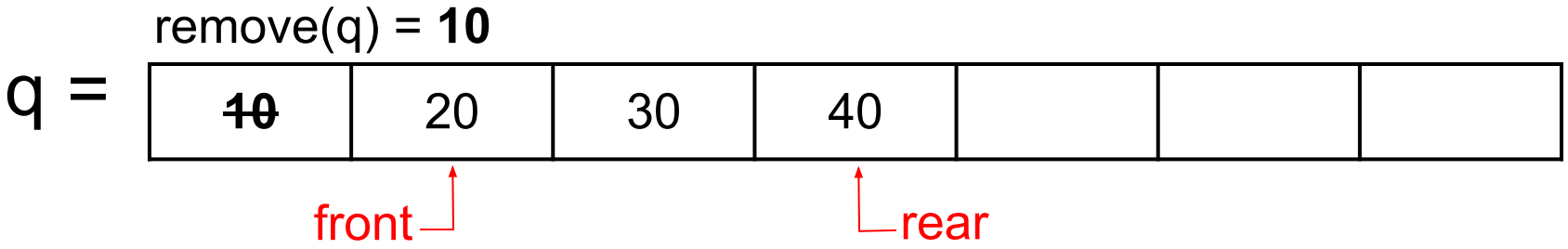
Exemplo de operações



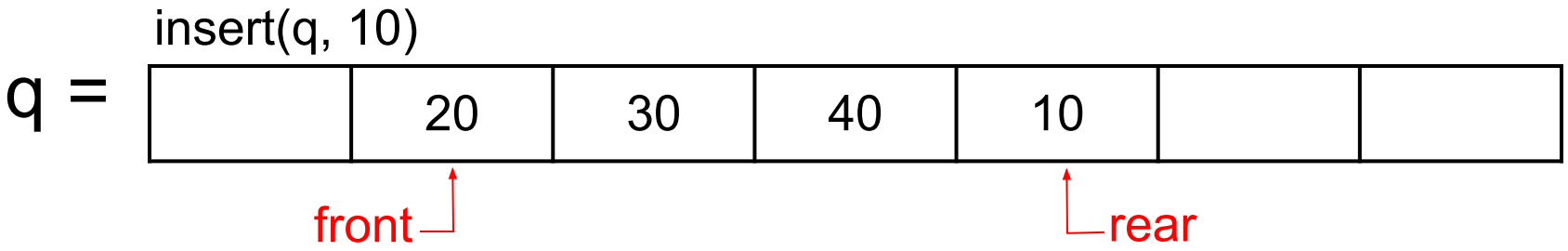
Exemplo de operações



Exemplo de operações



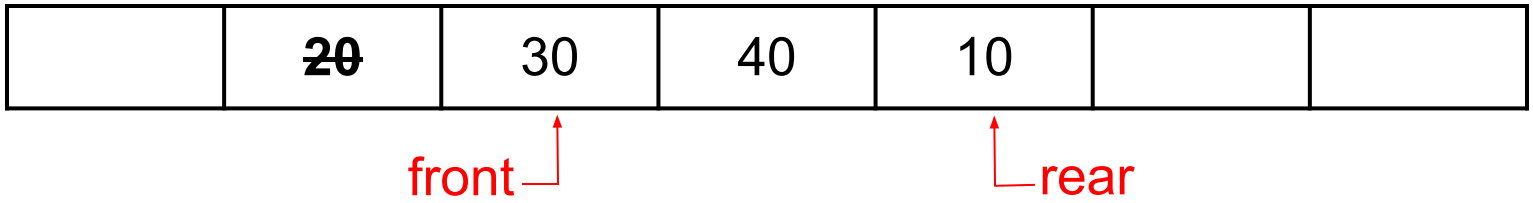
Exemplo de operações



Exemplo de operações

remove(q) = **20**

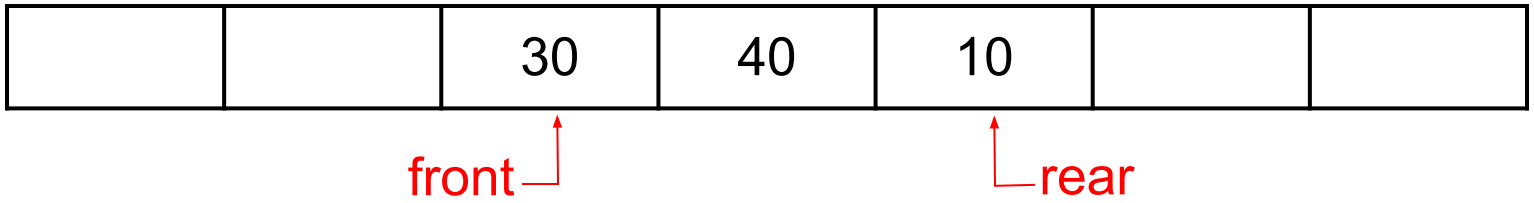
q =



Exemplo de operações

empty(q) = **false**

q =



Operação *insert*

- sempre pode ser executada:
  - teoricamente não há limite para o número de elementos que uma fila pode conter;
  - limitada pela memória disponível.

- erro de ***overflow***:
  - é o resultado de uma tentativa inválida de inserir um elemento em uma fila cheia.

Operação *remove*

- só pode ser aplicada se a fila **não estiver vazia**;
  - não existe como remover um elemento de uma fila vazia!

- erro ***underflow***:
  - é o resultado em uma tentativa de remover um elemento de uma fila vazia;
  - a operação ***empty*** é sempre aplicável.

Aspectos de Implementação usando Vetores

Aspectos de Implementação

- Como uma fila pode ser representada em C?
  - Uma ideia (**e menos indicada**) é usar **vetor** para armazenar os elementos da fila; e
  - duas variáveis, **front** e **rear**, para armazenar as posições dentro do vetor do primeiro e último elementos da fila.

Aspectos de Implementação

- Pode-se definir uma fila **q** de inteiros com:

  #define MAXQUEUE 100

Constante que define o tamanho máximo da fila

Aspectos de Implementação

- Pode-se definir uma fila **q** de inteiros com:

```
#define MAXQUEUE 100

typedef struct sQueue {
        int items[MAXQUEUE];
        int front, rear;
} Queue;
```

TAD da fila

Aspectos de Implementação

- Pode-se definir uma fila **q** de inteiros com:

```
#define MAXQUEUE 100

typedef struct sQueue {
        int items[MAXQUEUE];
        int front, rear;
} Queue;
```

TAD da fila

```
Queue q;
q.front = 0;
q.rear = -1;
```

Inicialização da fila

Aspectos de Implementação

- Inicialmente:
  - **q.rear** é definido com -1; e
  - **q.front** é definido com 0.

- A fila está vazia sempre que **q.rear < q.front**;

- O número de elementos na fila, a qualquer momento, é igual ao valor de **q.rear - q.front + 1**.

Exemplo de operações

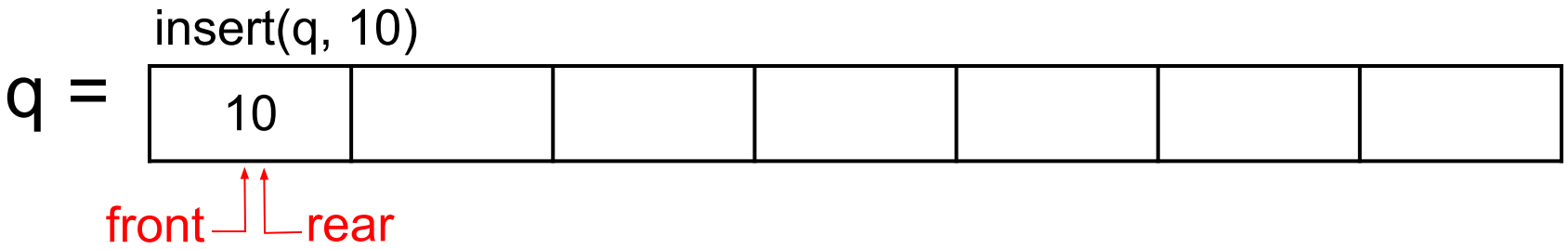
empty(q) = **true**

q = 

front

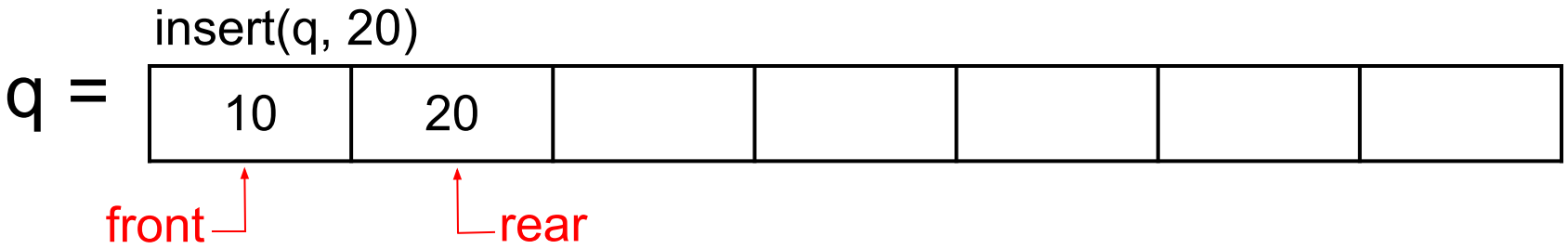
q.front = 0
q.rear = -1
size (q.rear - q.front + 1) = **-1 - 0 + 1 = 0**

Exemplo de operações



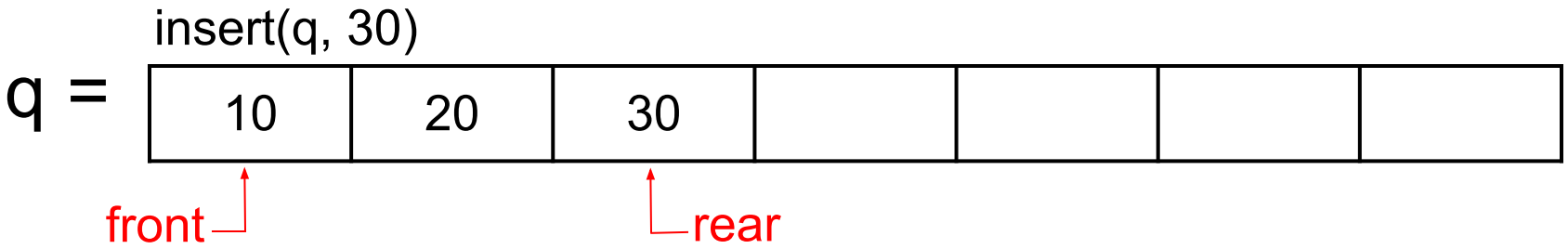
q.front = 0
q.rear = 0
size (q.rear - q.front + 1) = **0 - 0 + 1 = 1**

Exemplo de operações



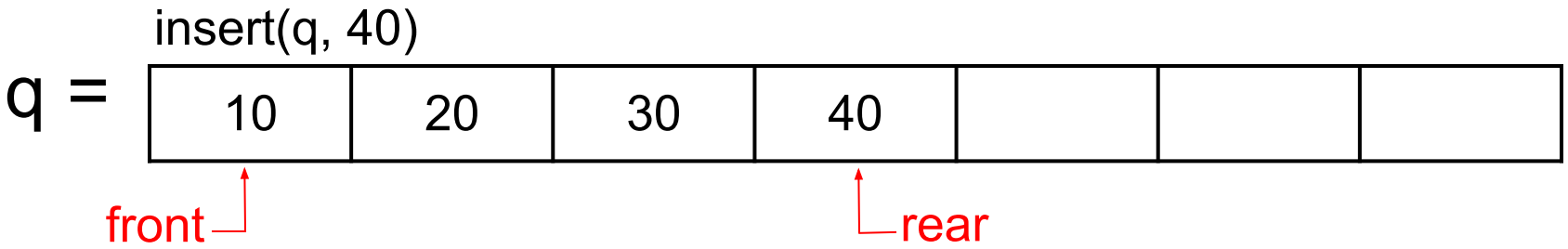
q.front = 0
q.rear = 1
size (q.rear - q.front + 1) = **1 - 0 + 1 = 2**

Exemplo de operações



q.front = 0
q.rear = 2
size (q.rear - q.front + 1) = **2 - 0 + 1 = 3**

Exemplo de operações

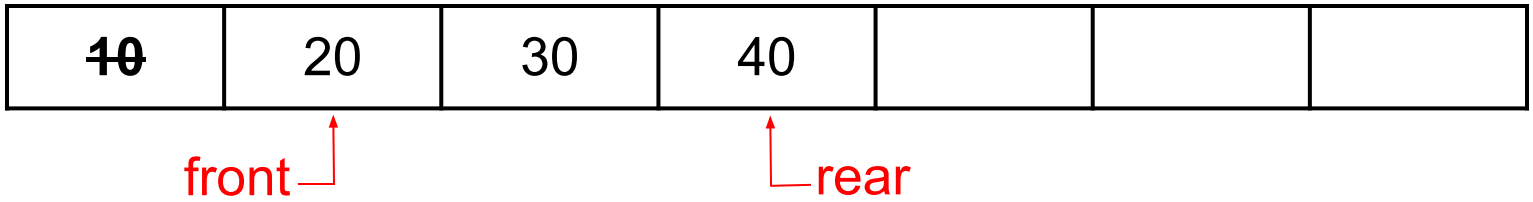


q.front = 0
q.rear = 3
size (q.rear - q.front + 1) = **3 - 0 + 1 = 4**

Exemplo de operações

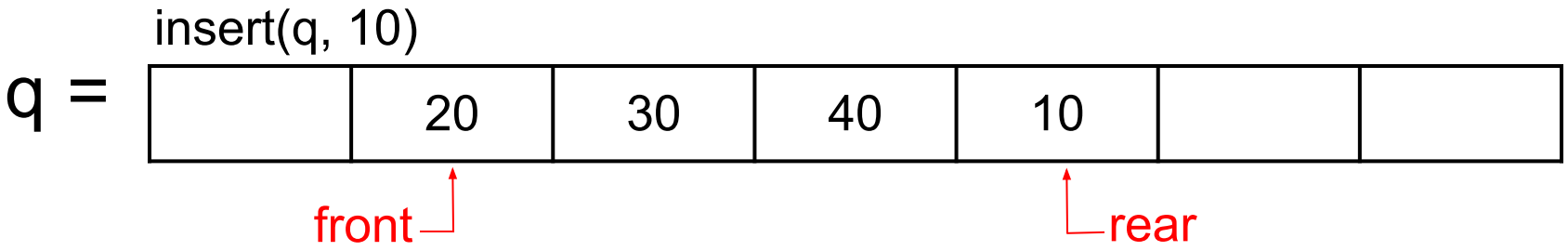
remove(q) = **10**

q =



q.front = 1
q.rear = 3
size (q.rear - q.front + 1) = **3 - 1 + 1 = 3**

Exemplo de operações

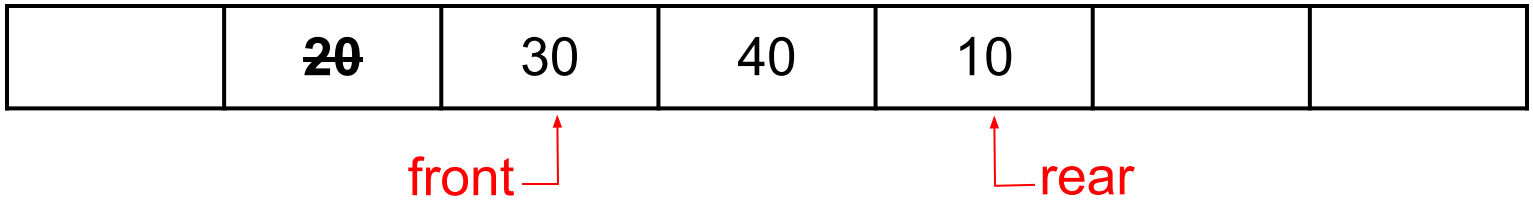


q.front = 1
q.rear = 4
size (q.rear - q.front + 1) = **4 - 1 + 1 = 4**

Exemplo de operações

remove(q) = **20**

q =



q.front = 2
q.rear = 4
size (q.rear - q.front + 1) = **4 - 2 + 1 = 3**

Aspectos de Implementação

Evidentemente, usar um vetor para armazenar uma fila introduz a possibilidade de "estouro" (***overflow***) caso a fila fique maior que o tamanho do vetor;

Aspectos de Implementação

- Ignorando a possibilidade de ***underflow*** e ***overflow***, a operação ***insert*(q, x)** poderia ser implementada pelas instruções:

  q.items [++q.rear] = x;

  - onde **x** é o elemento a ser inserido na fila.
- e a operação ***remove*(q)** poderia ser implementada por:

  x = q.items [q.front++];

  - onde **x** é o elemento removido e retornado.

Aspectos de Implementação usando Lista Encadeada

Aspectos de Implementação usando Lista Encadeada

- Pode ser implementada usando lista encadeada simples ou dupla;

- O ponteiro ***head*** pode substituir a variável ***front*** e o ponteiro ***tail*** substitui a variável ***rear***;

Aspectos de Implementação usando Lista Encadeada

- A operação ***insert()*** poderá ser implementada utilizando a inserção com elemento pivô ***tail*****:**
  - assim a inserção na fila se dará sempre pelo elemento ***tail***;

Aspectos de Implementação usando Lista Encadeada

- A operação ***remove()*** poderá ser implementada utilizando a remoção com elemento pivô **NULL**
  - ou pelo ***head*** no caso de lista duplamente encadeada;
  - a remoção sempre se dará no início da lista eliminando o elemento ***head***.

Aspectos de Implementação usando Lista Encadeada

- A operação ***empty()*** poderá ser implementada retornando **verdadeiro** caso o *size* da lista seja 0
  - ou **falso** caso seja maior que zero.

Aspectos de Implementação usando Lista Encadeada

- O erro de ***overflow*** irá acontecer somente quando não existir mais memória disponível para alocação;
- Por outro lado, o erro de ***underflow*** acontecerá sempre que se tentar remover um item em uma fila vazia.

Fila de Prioridade

Agenda

Aula anterior

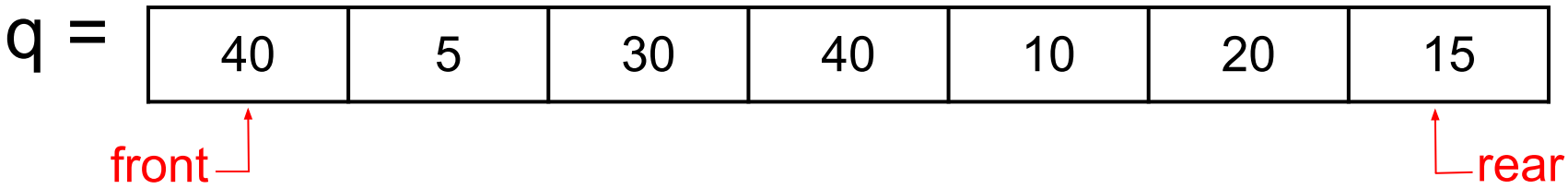
- Conceitos
- Aspectos de Implementação usando vetores
- Aspectos de Implementação usando Lista Encadeada
- Fila de Prioridades

Aula de hoje!

Filas tradicionais

- Recapitulando filas tradicionais:
  - as **filas** são **estruturas de dados** cujos elementos estão **ordenados** com base na **sequência** na qual foram inseridos;
  - a operação ***remove()*** recupera o primeiro elemento inserido.
- Se existir uma ordem intrínseca entre os próprios elementos (por exemplo, ordem numérica ou alfabética), **ela será ignorada nas operação da fila**.

Filas tradicionais



Fila de prioridade

- A **fila de prioridade** é uma **estrutura de dados** na qual a classificação intrínseca dos elementos determina os resultados de suas operações básicas.

- São basicamente de dois tipos:
  - ascendente; ou
  - descendente.

Conceito

Uma **fila de prioridade ascendente** é um tipo de fila na qual podem ser inseridos itens arbitrariamente e a partir da qual **apenas o menor item pode ser removido**.

Conceito

- Seja **apq** uma fila de prioridade ascendente, então:
  - a operação **pqInsert(apq, x)** insere o elemento **x** em **apq;**
  - a operação **pqMinRemove(apq)** remove o menor elemento de **apq** e retorna seu valor.

Conceito

- A operação **pqMinRemove(apq)** recupera sucessivamente elementos de uma fila de prioridade em ordem ascendente:
  - se um elemento pequeno for inserido depois de várias eliminações, a próxima recuperação retornará esse elemento pequeno
    - este elemento pode ser menor que um elemento anteriormente recuperado.

Conceito

- De modo semelhante, **pqMaxRemove** recupera elementos de uma fila de prioridade descendente, em **ordem descendente**:
  - Isso explica a designação de uma fila de prioridade como *ascendente* ou *descendente*;
  - As operações **pqMinRemove** ou **pqMaxRemove** só podem ser aplicadas a uma fila de prioridade não-vazia [isto é, se **empty(pq)** for **falso**];
  - a operação **empty(pq)** aplica-se da mesma forma que nas filas tradicionais.

Resumo

- Conceitos
- Aspectos de Implementação usando Vetores
- Aspectos de Implementação usando Lista Encadeada
- Fila de Prioridades

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CATARINENSE
Campus Videira

Filas

Professor: Manassés Ribeiro

manasses.ribeiro@ifc.edu.br