Estrutura de Dados

Aula 06

Prof. Luiz Antonio Schalata Pacheco, Dr. Eng.

Instituto Federal de Santa Catarina
Câmpus Garopaba
Curso Superior de Tecnologia em Sistemas para Internet

schalata@ifsc.edu.br

30/03/2023

Filas

Conceito

- Representam as filas do mundo real

Conceito

- As pessoas que chegam primeiro numa fila são as que serão atendidas antes. Excluindo-se as prioridades (veremos depois!)
- Uma fila é uma lista linear de informações, que é acessada na ordem primeiro que chega é o primeiro que sai
- *FIFO - First In, First Out*
- O primeiro item da fila é o primeiro a ser recuperado, o segundo item colocado é o segundo a ser recuperado e assim por diante
- Não são permitidos acessos randômicos a nenhum item específico

Operações

- Enfileirar: colocar um item no final da fila
- Desenfileirar: remover um item do início da fila
- Mostra elemento: ver início da fila

Funcionamento

Ação	Conteúdo da fila
enfileirar(A)	A
enfileirar(B)	AB
enfileirar(C)	ABC
desenfileirar()	BC
enfileirar(D)	BCD
desenfileirar()	CD
desenfileirar()	D

Aplicações

- Filas são usadas em muitas situações de programação. Umas das mais comuns é um simulação
- Modelar aviões aguardando para decolar num aeroporto
- Modelar pacotes de dados para serem transmitidos pela rede (algoritmos baseados em filas são muito utilizados em SO e redes de computadores)
- Fila de impressora, na qual os serviços de impressão aguardam a impressora ficar disponível

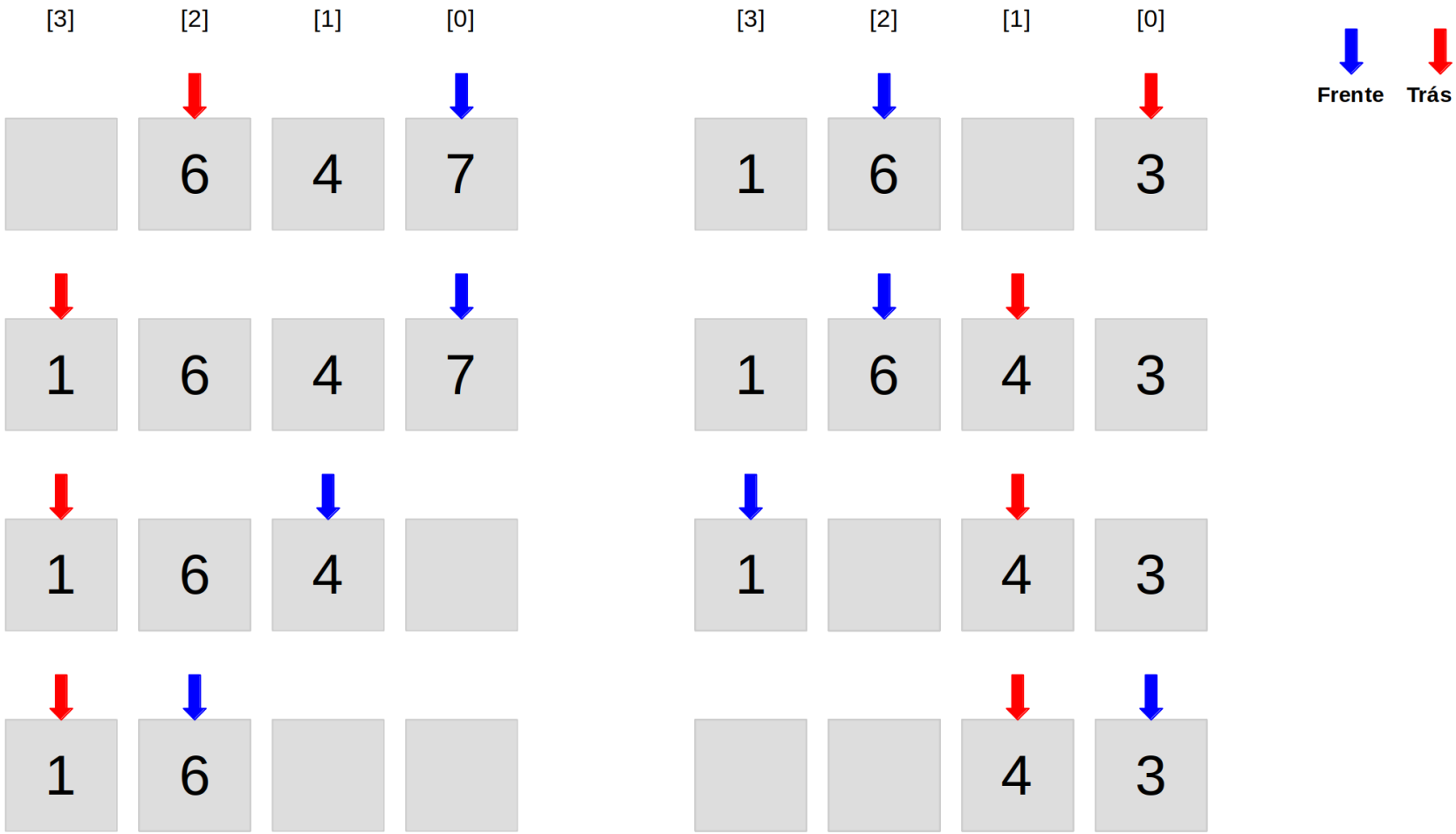
Implementação

```
queue = []  # Cria a fila
queue.append("A")  # Metodo enfileirar()
print(queue[0])  # Metodo para mostrar elemento()
queue.append("B")
print(queue[0])
queue.append("C")
print(queue[0])
print(queue.pop(0))  # Metodo desenfileirar()
print(queue[0])
```

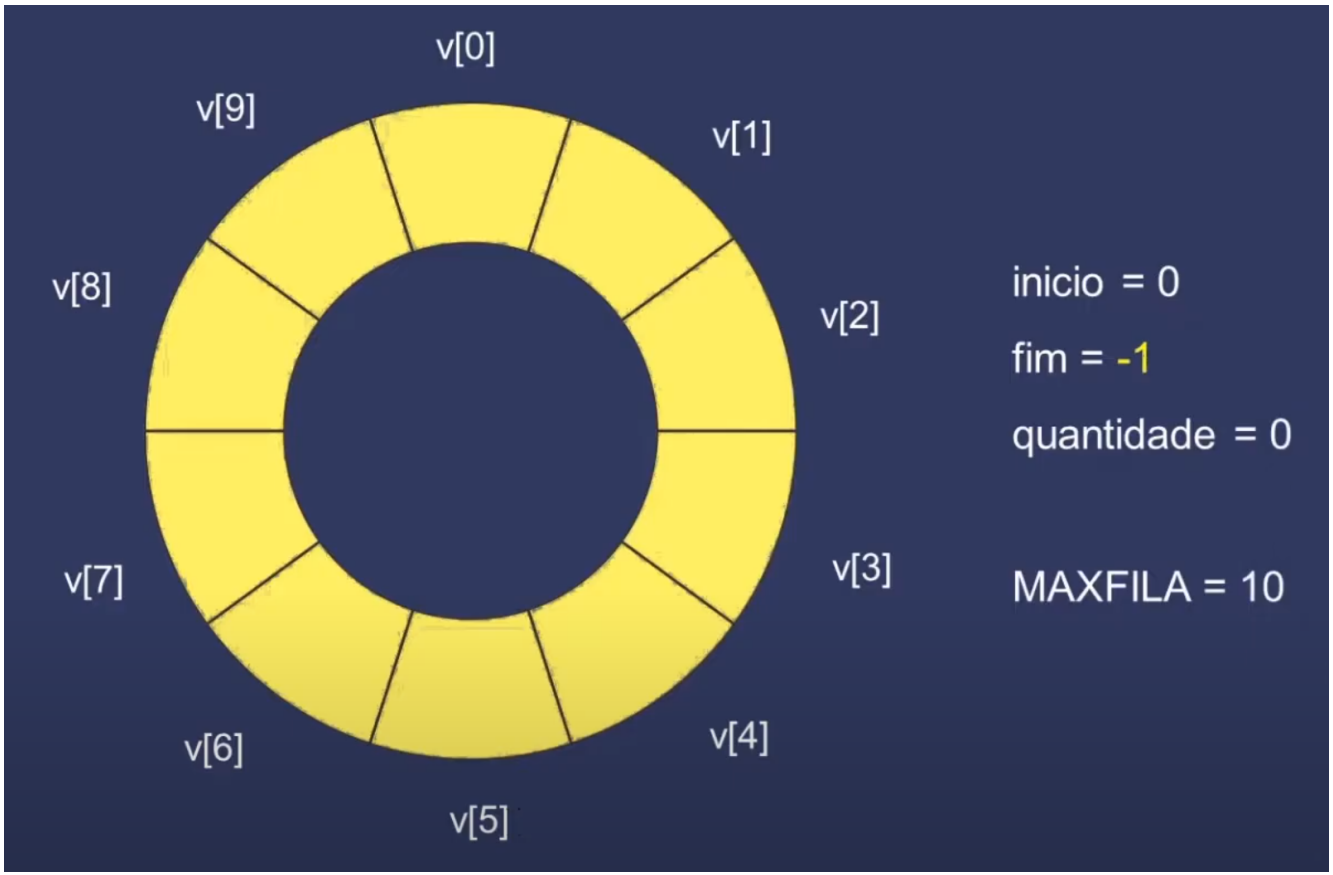

Implementação

- Porém, filas não devem ser implementadas a partir de listas:
  - A operação desenfileirar() implica na necessidade de fazer o remanejamento dos elementos
  - Big(O) = O(n)
  - Utiliza-se a ideia de fila circular

Fila circular com vetor



Fila circular com vetor



Fila circular

- Não há movimentação efetiva dos dados
- Com isso não há necessidade de fazer o remanejamento dos elementos
- Os elementos circulam dentro do vetor
- É necessário o registro da quantidade atual de elementos
- Além do fim, é preciso manter a posição de início da fila
- Fazer uma exclusão, num vetor ordenado ou não ordenado, tem complexidade O(n), na fila circular passa a ser O(1), portanto constante.

Implementação

- Implementar uma Fila Circular a partir de um vetor com os respectivos métodos de enfileirar, desenfileirar e ver início da fila.
- Além disso faça duas funções adicionais para verificar se a fila está cheia e se a fila está vazia.

Fila circular com vetor: implementação

```
1  import numpy as np
2
3  class FilaCircular:
4
5      def __init__(self, capacidade):
6          self.capacidade = capacidade
7          self.inicio = 0  # Indica o inicio
8          self.final = -1  # Indica o fim
9          self.numero_elementos = 0  # Qtidade de elementos
10         self.valores = np.empty(self.capacidade, dtype=
       int)
11
12     def __fila_vazia(self):
13         return self.numero_elementos == 0
14
15     def __fila_cheia(self):
16         return self.numero_elementos == self.capacidade
```


Fila circular: implementação (continuação)

```
18      def enfileirar(self, valor):
19          if self.__fila_cheia():
20              print('A fila esta cheia')
21              return
22
23          # Se o final da fila chegou no final do vetor,
24          # deve voltar na posicao 0
25          self.final = (self.final + 1) % self.capacidade
26          self.valores[self.final] = valor
27          self.numero_elementos += 1
```


Fila circular: implementação (continuação)

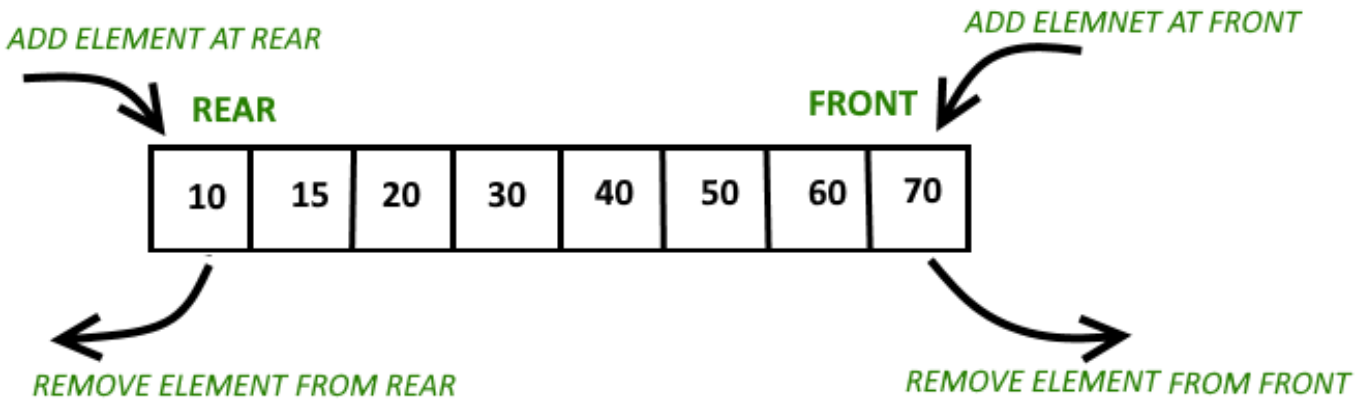
```
29      def desenfileirar(self):
30          if self.__fila_vazia():
31              print('A fila ja esta vazia')
32              return
33
34          dado = self.valores[self.inicio]
35          self.inicio = (self.inicio + 1) % self.capacidade
36          self.numero_elementos -= 1
37          return dado
38
39      def retorna_primeiro(self):
40          if self.__fila_vazia():
41              return -1
42          return self.valores[self.inicio]
```


Deques

Deques (Double Ended Queue)

- Combinação de Pilhas e Filas
- Quatro operações
  - Adicionar na frente
  - Remover na frente
  - Adicionar no final
  - Remover no final
- Implementação estática ou circular (mais recomendável)
- Aplicações: filas com acesso prioritário onde a maioria das entidades segue a lógica padrão de fila, mas existem casos prioritários que serão empurrados para a frente desta fila

Deque



Fonte: https://www.geeksforgeeks.org/implementation-deque-using-circular-array/

Deques: implementação

```
1  import numpy as np
2
3  class Deque:
4
5    def __init__(self, capacidade):
6      self.capacidade = capacidade
7      self.inicio = -1
8      self.final = 0
9      self.numero_elementos = 0
10     self.valores = np.empty(self.capacidade, dtype=int)
11
12   def __deque_cheio(self):
13     return (self.inicio == 0 and self.final == self.
       capacidade - 1) or (self.inicio == self.final + 1)
14
15   def __deque_vazio(self):
16     return self.inicio == -1
```


Deques: Função para inserir no início

```
18   def insere_inicio(self, valor):
19     if self.__deque_cheio():
20       print('O deque esta cheio')
21       return
22
23     # Se estiver vazio
24     if self.inicio == -1:
25       self.inicio = 0
26       self.final = 0
27     # Inicio estiver na primeira posicao
28     elif self.inicio == 0:
29       self.inicio = self.capacidade - 1
30     else:
31       self.inicio -= 1
32
33     self.valores[self.inicio] = valor
```


Deques: Função para inserir no final

```
35    def insere_final(self, valor):
36      if self.__deque_cheio():
37        print('O deque esta cheio')
38        return
39
40      # Se estiver vazio
41      if self.inicio == -1:
42        self.inicio = 0
43        self.final = 0
44      # Final estiver na ultima posicao
45      elif self.final == self.capacidade - 1:
46        self.final = 0
47      else:
48        self.final += 1
49
50      self.valores[self.final] = valor
```

Deques: Função para excluir do início

```
52  def excluir_inicio(self):
53    if self.__deque_vazio():
54      print('O deque ja esta vazio')
55      return
56
57    # Possui somente um elemento
58    if self.inicio == self.final:
59      self.inicio = -1
60      self.final = -1
61    else:
62      # Volta para a posicao inicial
63      if self.inicio == self.capacidade - 1:
64        self.inicio = 0
65      else:
66        # Incrementar inicio para remover o inicio atual
67        self.inicio += 1
```


Deques: Função para excluir do final

```
69    def excluir_final(self):
70      if self.__deque_vazio():
71        print('O deque ja esta vazio')
72        return
73
74      if self.inicio == self.final:
75        self.inicio = -1
76        self.final = -1
77      elif self.inicio == 0:
78        self.final = self.capacidade - 1
79      else:
80        self.final -= 1
```


Deques: Retornar elemento do início e do final

```
82    def get_inicio(self):
83      if self.__deque_vazio():
84        print('O deque ja esta vazio')
85        return
86
87      return self.valores[self.inicio]
88
89    def get_final(self):
90      if self.__deque_vazio() or self.final < 0:
91        print('O deque ja esta vazio')
92        return
93
94      return self.valores[self.final]
```