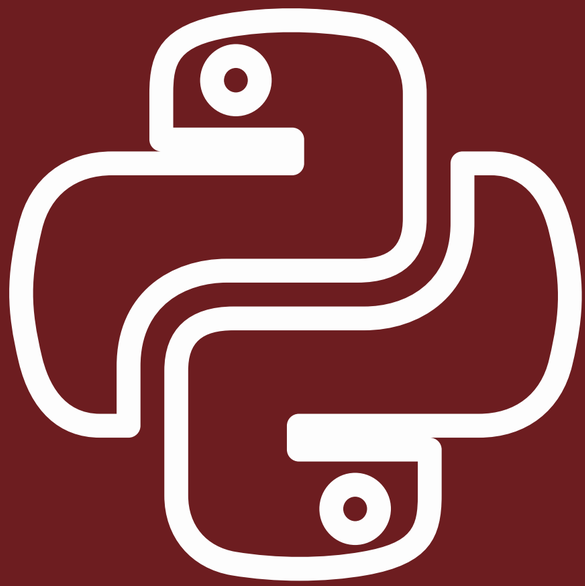
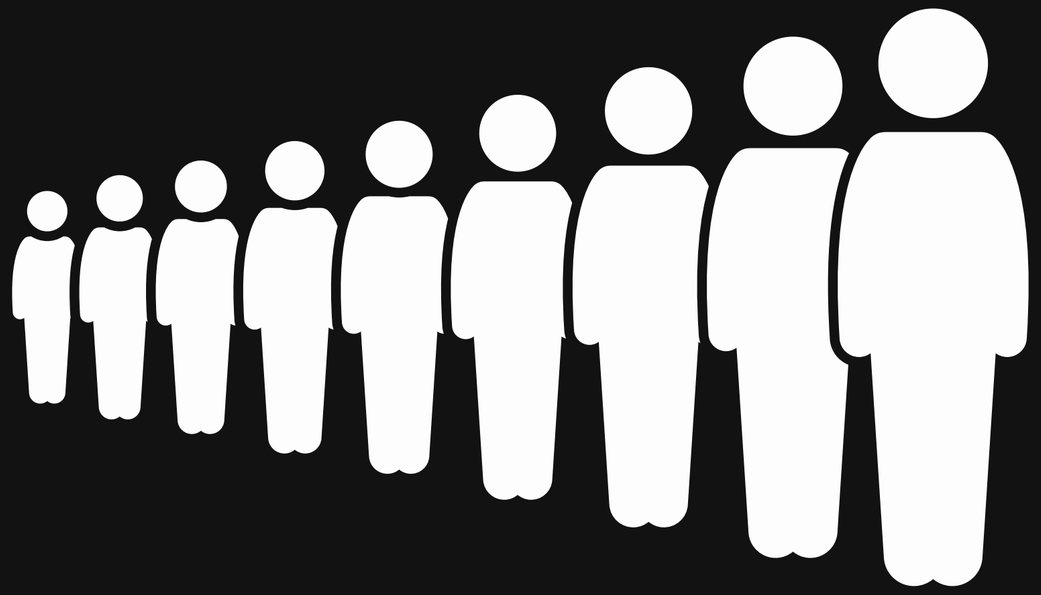
ESTRUTURA DE DADOS

UNIVERSIDADE DE
VASSOURAS



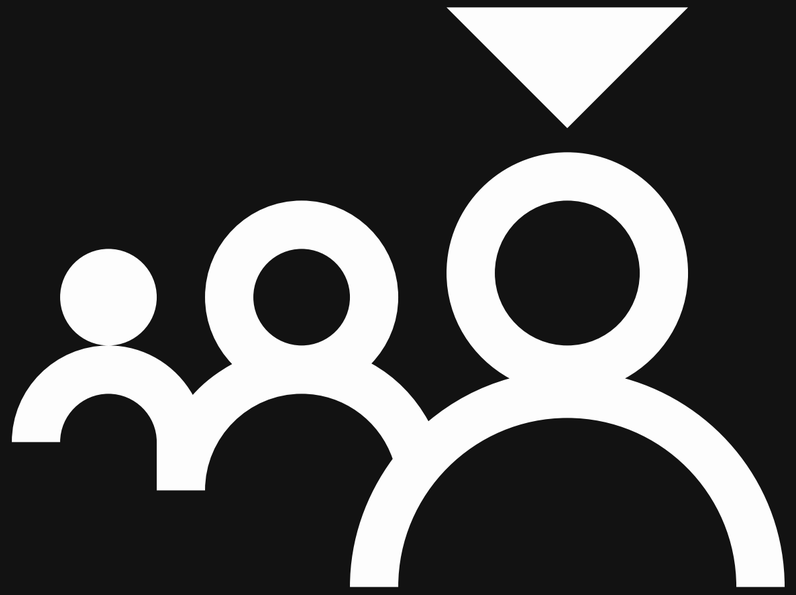
Prof. Me. Caio Jannuzzi

FILAS



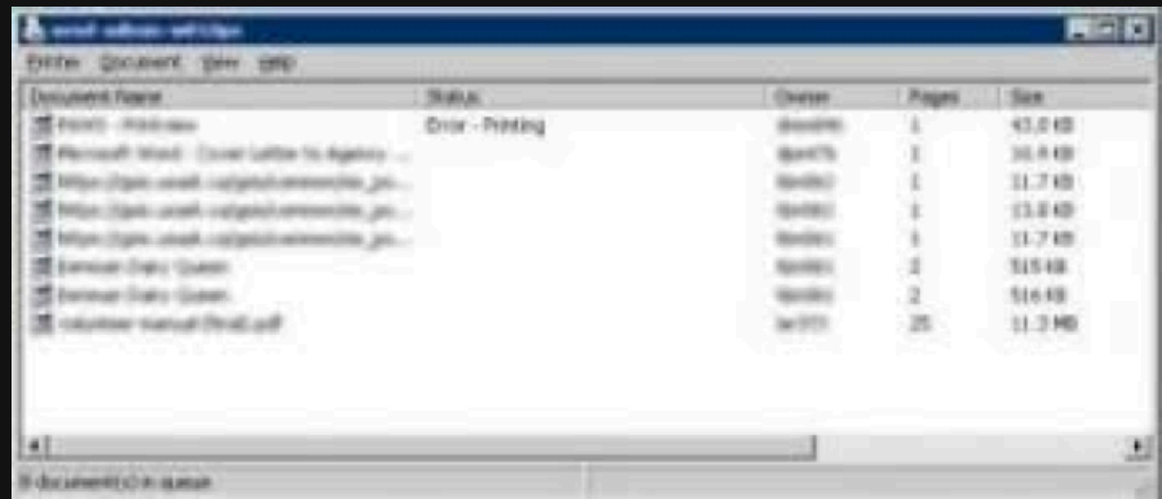
A PRIMEIRA PESSOA A ENTRAR NO FINAL DA FILA SERÁ A PRIMEIRA PESSOA A CHEGAR NA FRENTE DA FILA.

É UMA ESTRUTURA SEMELHANTE A UMA PILHA, EXCETO QUE EM UMA FILA O PRIMEIRO ELEMENTO INSERIDO É O PRIMEIRO A SER REMOVIDO (FIRST-IN- FIRST-OUT, FIFO – PRIMEIRO-A-ENTRAR-PRIMEIRO-A-SAIR)



APLICAÇÕES

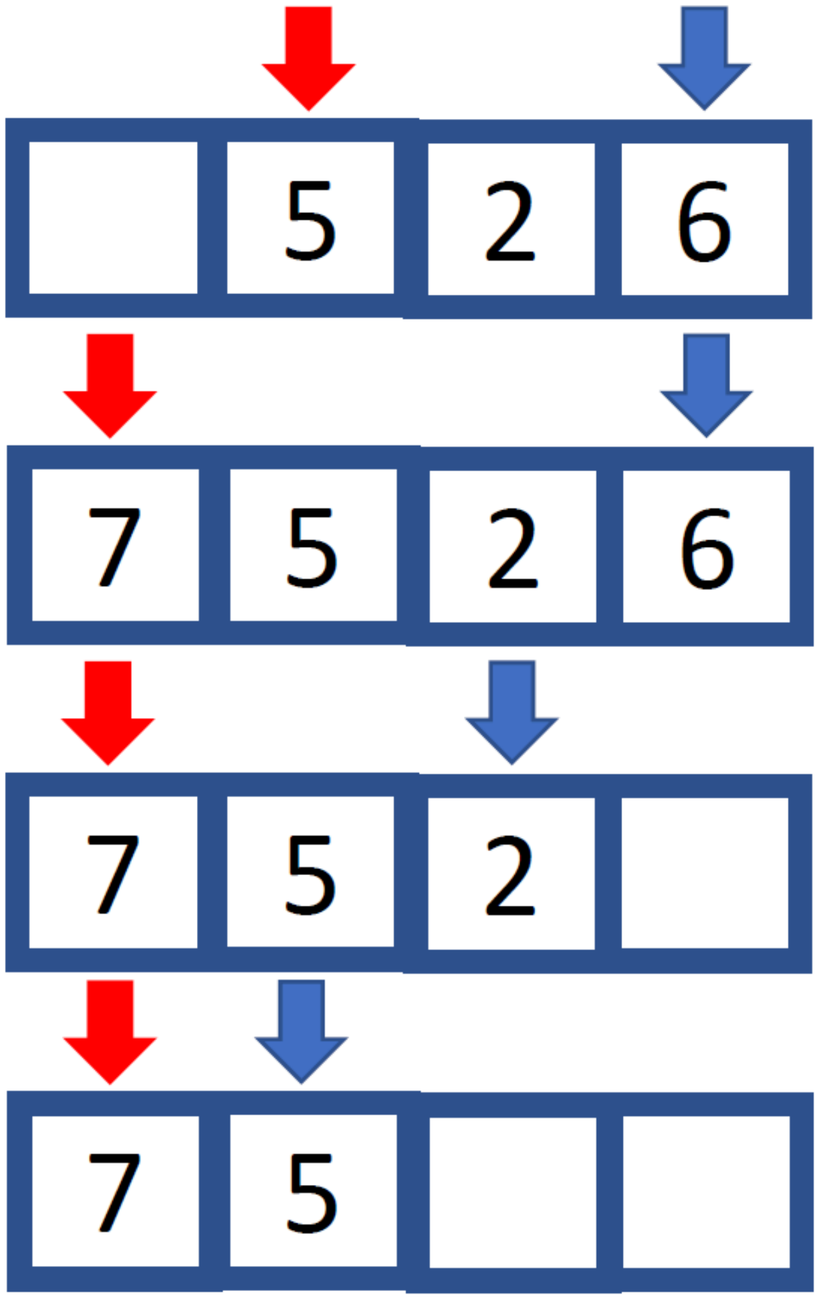
- MODELAR AVIÕES AGUARDANDO PARA DECOLAR
- PACOTES DE DADOS ESPERANDO PARA SEREM TRANSMITIDOS PELA REDE
- FILA DA IMPRESSORA, NO QUAL SERVIÇOS DE IMPRESSÃO AGUARDAM A IMPRESSORA FICAR DISPONÍVEL

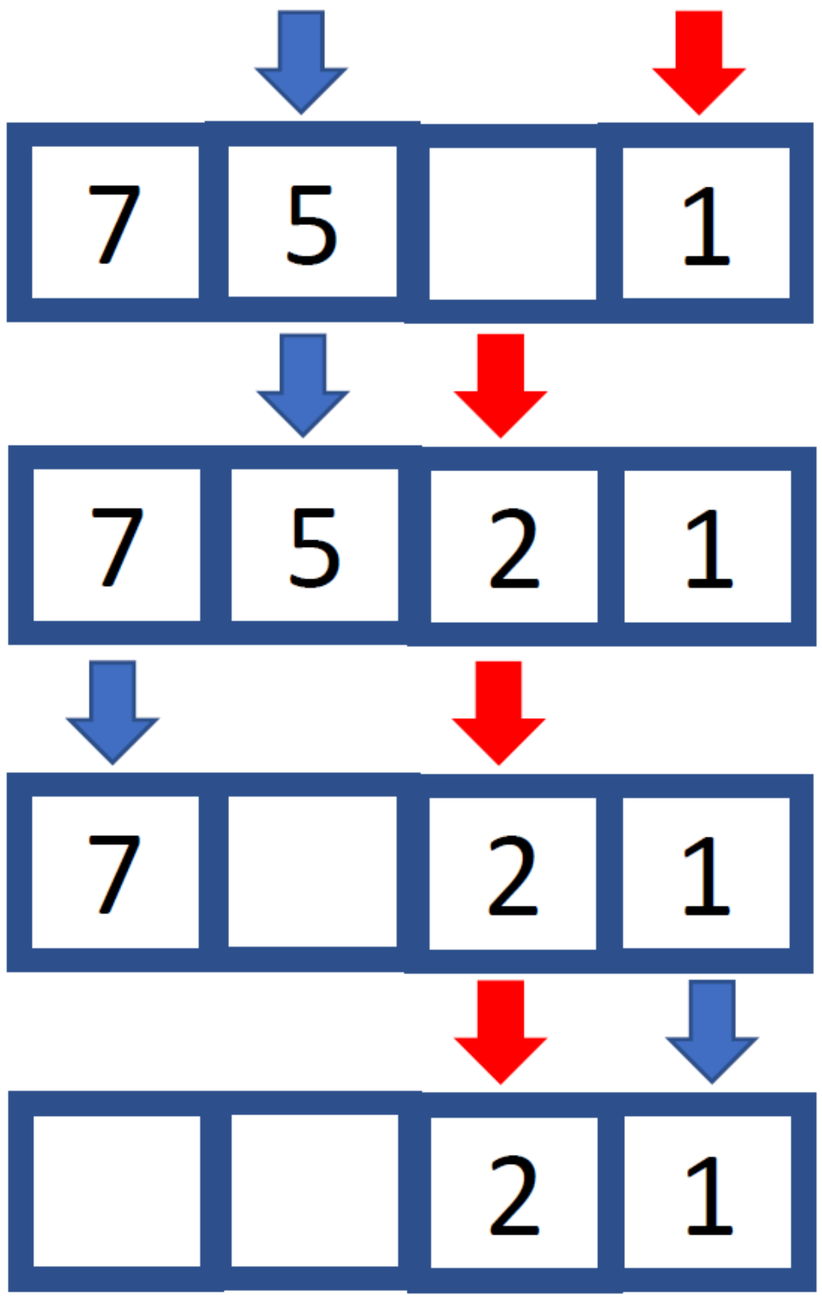


FILAS – OPERAÇÕES

- ENFILEIRAR
  - COLOCAR UM ITEM NO FINAL DA FILA
- DESENFILEIRAR
  - REMOVER UM ITEM DO INÍCIO DA FILA
- VER INÍCIO DA FILA
  - MOSTRA O ELEMENTO QUE ESTÁ NO INÍCIO DA FILA

FILA CIRCULAR





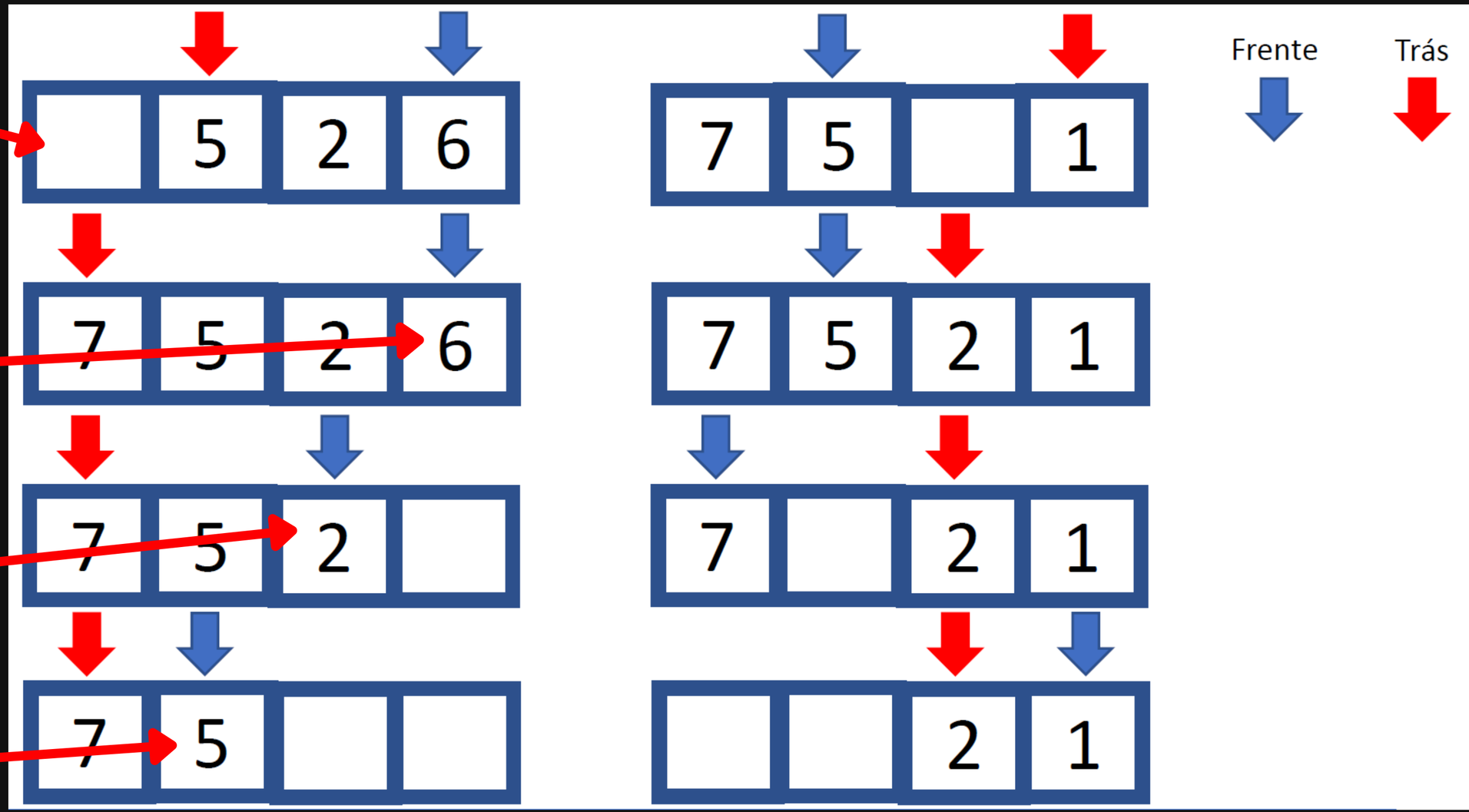
FILA CIRCULAR

Tem vaga vai para o final da fila

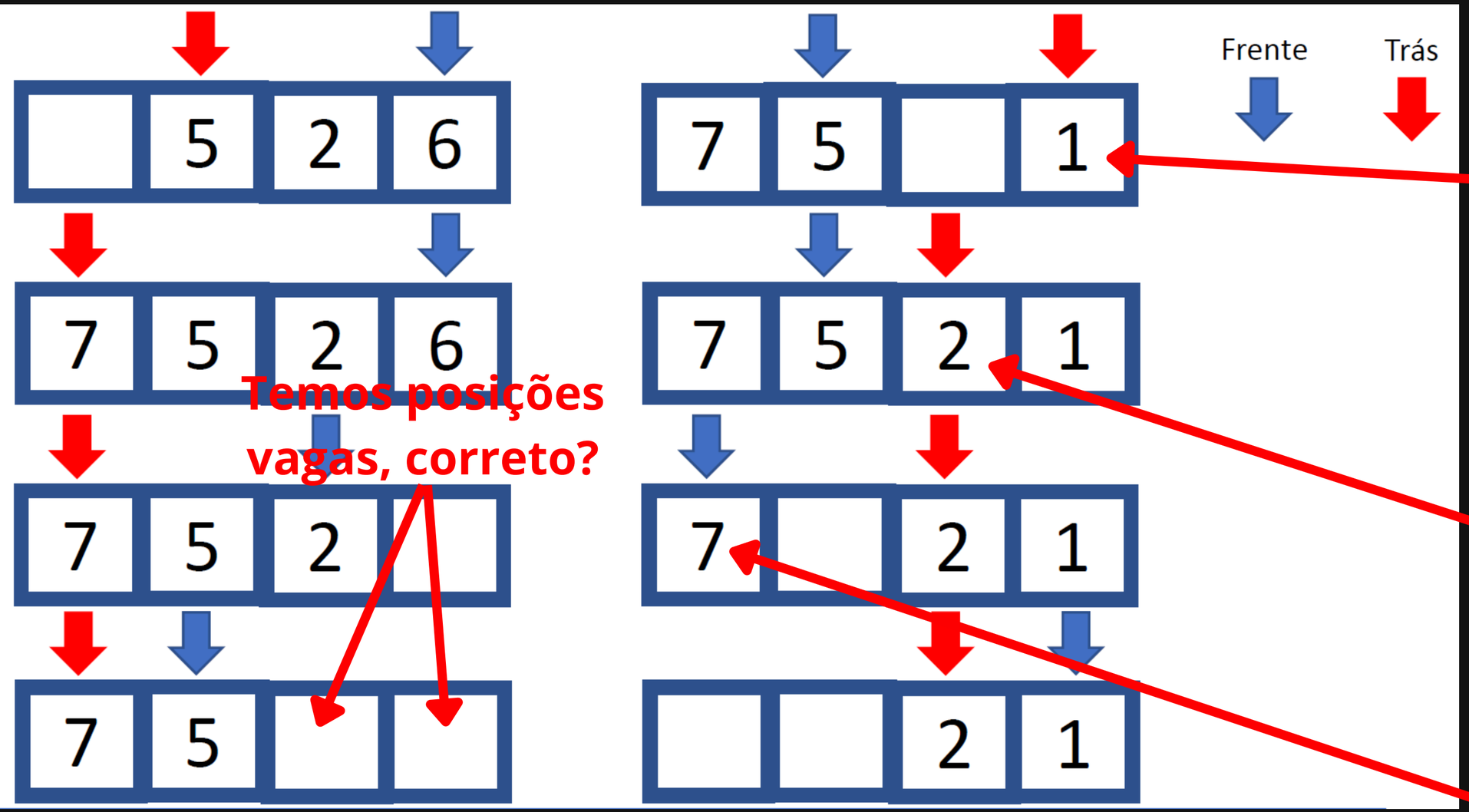
Primeiro a entrar é o primeiro a sair

Primeiro a entrar é o primeiro a sair

Primeiro a entrar é o primeiro a sair



FILA CIRCULAR



Então vamos inserir novos números nas posições vagas, mudando o cursor de final de fila.

A cada nova inserção deve-se atualizar o índice de final de fila.

A cada nova saída deve-se atualizar o índice de início de fila.

IMPLEMENTAÇÃO DE FILA CIRCULAR

Fila circular

```python
import numpy as np
```

[1] ✓ 0.0s

```python
class FilaCircular:

  def __init__(self, capacidade):
    self.capacidade = capacidade
    self.inicio = 0
    self.final = -1
    self.numero_elementos = 0
    self.valores = np.empty(self.capacidade, dtype=int)

```

```
10  def __fila_vazia(self): # O(1)
11      return self.numero_elementos == 0
12
13  def __fila_cheia(self):
14      return self.numero_elementos == self.capacidade # retorna True se a fila estiver
        cheia
15
16  def enfileirar(self, valor):
17      if self.__fila_cheia(): # Se a fila estiver cheia retorna True e encerra o programa
18          print('A fila está cheia')
19          return
20
21      if self.final == self.capacidade - 1: # Se o final estiver na ultima posicao
        retorna True e encerra o programa
22          self.final = -1 # Volta o final para a primeira posicao
23      self.final += 1 # Incrementa o final
24      self.valores[self.final] = valor # Adiciona o valor
25      self.numero_elementos += 1 # Incrementa o numero de elementos
26
```

```
27  def desenfileirar(self):
28      if self.__fila_vazia():
29          print('A fila já está vazia')
30          return
31
32      temp = self.valores[self.inicio] # Guarda o valor que vai ser removido na variável
        temp
33      self.inicio += 1 # Incrementa o inicio para a proxima posicao
34      if self.inicio == self.capacidade: # Se o inicio estiver na ultima posicao retorna
        True
35          self.inicio = 0 # Volta o inicio para a primeira posicao
36      self.numero_elementos -= 1 # Decrementa o numero de elementos
37      return temp
```

```
38
39  def primeiro(self): # Função que retorna o primeiro elemento da fila
40      if self.__fila_vazia(): # Se a fila estiver vazia retorna -1 e encerra o programa
41          return -1 # Retorna -1
42      return self.valores[self.inicio] # retorna o primeiro elemento da fila
43
```

```python
fila = FilaCircular(5)
```

[18] ✓ 0.0s Python

```python
fila.primeiro()
```

[19] ✓ 0.0s Python

```
-1
```

```python
# 1
fila.enfileirar(1)
fila.primeiro()
```

[20] ✓ 0.0s Python

```
1
```

```python
# 2 1
fila.enfileirar(2)
fila.primeiro()
```

[21] ✓ 0.0s Python

```
1
```

```python
# 5 4 3 2 1
fila.enfileirar(3)
fila.enfileirar(4)
fila.enfileirar(5)
```

[22] ✓ 0.0s Python

```python
fila.enfileirar(6)
```

[23] ✓ 0.0s Python

```
A fila está cheia
```

```python
# 5 4 3
fila.desenfileirar()
fila.desenfileirar()
fila.primeiro()
```

[24] ✓ 0.0s Python

```
3
```

```python
# 7 6 5 4 3
fila.enfileirar(6)
fila.enfileirar(7)
```

[25] ✓ 0.0s Python

```python
fila.primeiro()
```

[26] ✓ 0.0s Python

```
3
```

```python
fila.valores
```

[27] ✓ 0.0s Python

```
array([6, 7, 3, 4, 5])
```

```
1 fila.inicio, fila.final
```

[28] ✓ 0.0s Python

```
(2, 1)
```

```
1 fila.valores[fila.final]
```

[29] ✓ 0.0s Python

```
7
```

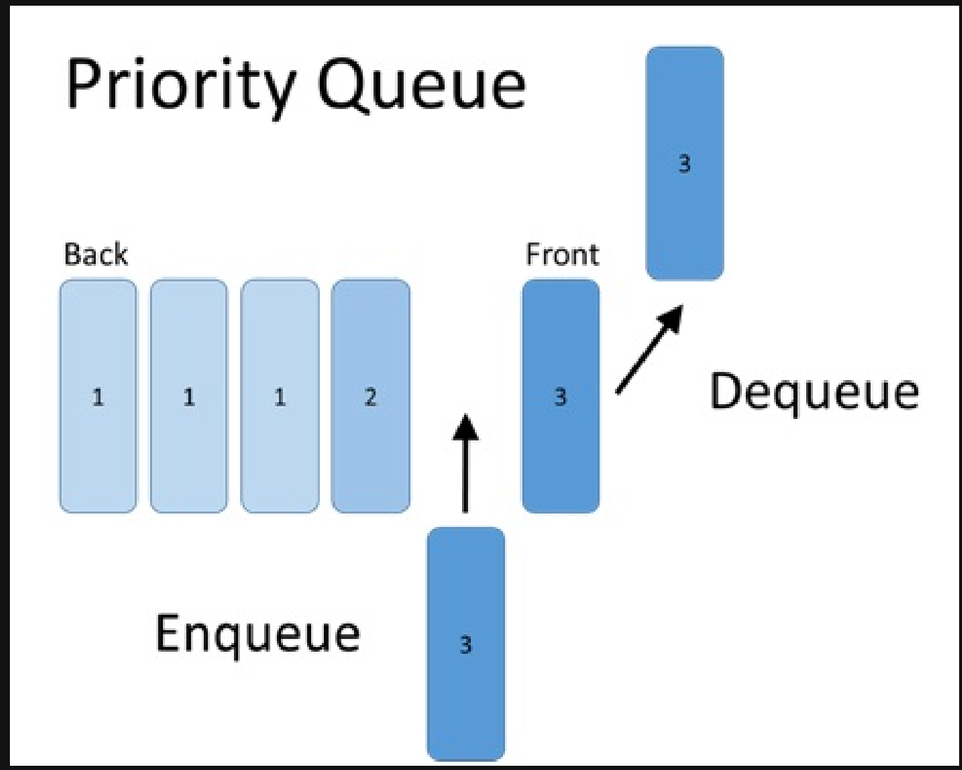
```
1 fila.valores[fila.inicio]
```

[30] ✓ 0.0s Python

```
3
```

FILA DE PRIORIDADE

- OS ITENS SÃO ORDENADOS POR VALOR-CHAVE, DE MODO QUE O ITEM COM A CHAVE MAIS BAIXA/ALTA ESTEJA SEMPRE NA FRENTE
- ELEMENTOS DE ALTA PRIORJDADE SÃO COLOCADOS NO INÍCIO DA FILA, DE MÉDIA PRIORIDADE NO MEIO DA FILA E ELEMENTOS DE BAIXA PRIORIDADE NO FINAL DA FILA



FILA DE PRIORIDADE

O maior na frente/ atrás

O menor vai sendo ordenado atrás do maior/ menor sempre .

FILA DE PRIORIDADE - EXEMPLO

VAMOS SUPOR QUE EXISTE UM DETERMINADO FUNCIONÁRIO QUE TENHA PRIORIDADE SOBRE O USO DA IMPRESSORA E UTILIZE A IMPRESSORA MAIS VEZES DO QUE OS OUTROS FUNCIONÁRIOS. É IMPORTANTE PARA ESSE FUNCIONÁRIO QUE AS IMPRESSÕES DELE SEJAM FEITAS PRIMEIRO. ÈLE PODE TER UM NÍVEL DE PRIORIDADE INDEPENDENTE SE TIVER DEZ ELEMENTOS NA FILA 10 PESSOAS QUE MANDARAM O DOCUMENTO PARA IMPRESSÃO ASSIM QUE ESSE FUNCIONÁRIO CLICA NO BOTÃO IMPRIMIR O DOCUMENTO QUE ELE MANDOU OU VAI PARA O INÍCIO DA FILA. QUER DIZER QUE A PRIORIDADE DELE SERÁ MAIOR DO QUE O DOS OUTROS FUNCIONÁRIOS. AQUI NÓS TEMOS UM EXEMPLO DE UMA FILA DE PRIORIDADE.

IMPLEMENTAÇÃO DE FILA DE PRIORIDADE

Fila de prioridade

```
[20]  ✓ 0.0s
```

```python
import numpy as np
```

```python
class FilaPrioridade:

    def __init__(self, capacidade):  # método construtor
        self.capacidade = capacidade
        self.numero_elementos = 0
        self.valores = np.empty(self.capacidade, dtype=int)

    def __fila_vazia(self):  # método auxiliar
        return self.numero_elementos == 0  # verifica se a fila esta vazia

    def __fila_cheia(self):  # método auxiliar
        return self.numero_elementos == self.capacidade  # verifica se a fila esta
        cheia

```

```
14      def enfileirar(self, valor):
15          if self.__fila_cheia():
16              print("A fila está cheia")
17              return
18
19          if self.numero_elementos == 0:  # primeiro elemento
20              self.valores[self.numero_elementos] = valor
21              self.numero_elementos += 1  # incrementa o número de elementos na fila
22          else:
23              x = self.numero_elementos - 1  # posição do elemento
24              while x >= 0:  # busca binária
25                  if valor > self.valores[x]:  # elemento é maior
26                      self.valores[x + 1] = self.valores[
27                          x
28                      ]  # desloca para a direita o elemento para abrir espaço para o
                           novo elemento
29                  else:
30                      break  # sai do loop quando o elemento for menor que o valor
                           passado como parametro
31                  x -= 1  # decrementa a posição do elemento
32              self.valores[x + 1] = valor  # coloca o elemento na nova posição
33              self.numero_elementos += 1  # incrementa o número de elementos na fila
34
```

```
34
35      def desenfileirar(self):  # remover o primeiro elemento da fila
36          if self.__fila_vazia():  # se a fila estiver vazia
37              print("A fila está vazia")  # imprime a mensagem
38              return
39
40          valor = self.valores[
41              self.numero_elementos - 1
42          ]  # pega o valor do último elemento
43          self.numero_elementos -= 1  # decrementa o número de elementos na fila
44          return valor
45
46      def primeiro(self):
47          if self.__fila_vazia():  # se a fila estiver vazia
48              return -1
49          return self.valores[self.numero_elementos - 1]
```

✓ 0.0s

```python
fila = FilaPrioridade(5)
```

[22] ✓ 0.0s Python

```python
fila.primeiro()
```

[23] ✓ 0.0s Python

```
-1
```

```python
# 30
fila.enfileirar(30)
fila.primeiro()
```

[24] ✓ 0.0s Python

```
30
```

```python
# 50 30
fila.enfileirar(50)
fila.primeiro()
```

[25] ✓ 0.0s Python

```
30
```

```
1  # Tradicional: 10 50 30
2  # Prioridade: 50 30 10
3  fila.enfileirar(10)
4  fila.primeiro()
```

[26] ✓ 0.0s

```
10
```

```
1  fila.valores
```

[27] ✓ 0.0s

```
array([ 50,  30,  10, 559,   0])
```

```
1  # Tradicional: 40 10 50 30
2  # Prioridade: 50 40 30 10
3  fila.enfileirar(40)
4  fila.primeiro()
```

[28] ✓ 0.0s

```
10
```

```
1  fila.valores
```

[29] ✓ 0.0s

```
array([50, 40, 30, 10,  0])
```

```
1  # Tradicional: 20 40 10 50 30
2  # Prioridade: 50 40 30 20 10
3  fila.enfileirar(20)
4  fila.primeiro()
```

[30] ✓ 0.0s

```
10
```

```
1  fila.valores
```

[31] ✓ 0.0s

```
array([50, 40, 30, 20, 10])
```

```
1  fila.enfileirar(2)
```

[32] ✓ 0.0s

```
A fila está cheia
```

```
1  fila.desenfileirar()
2  fila.primeiro()
```

[33] ✓ 0.0s

```
20
```

```
1  fila.desenfileirar()
2  fila.primeiro()
```

[34] ✓ 0.0s

```
30
```

```
1  fila.desenfileirar()
2  fila.primeiro()
```

[35] ✓ 0.0s

```
40
```

```
1  fila.valores
```

[36] ✓ 0.0s

```
array([50, 40, 30, 20, 10])
```

```
1  fila.enfileirar(5)
2  fila.primeiro()
```

[37] ✓ 0.0s

```
5
```

```
1  fila.valores
```

[38] ✓ 0.0s

```
array([50, 40,  5, 20, 10])
```


That's all Folks!
ATÉ A PRÓXIMA!