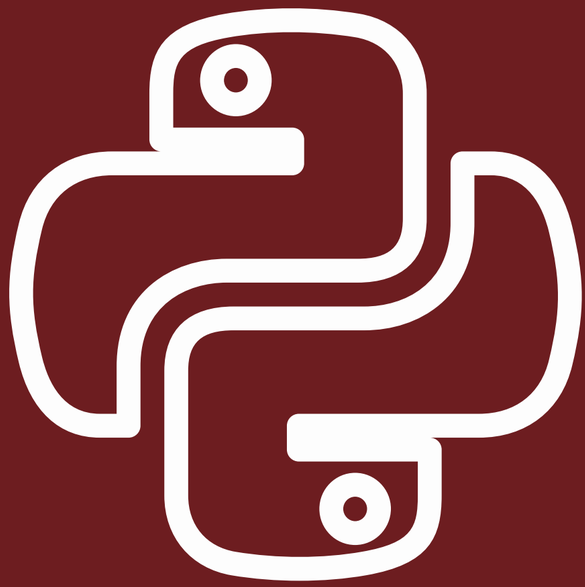
ESTRUTURA DE DADOS

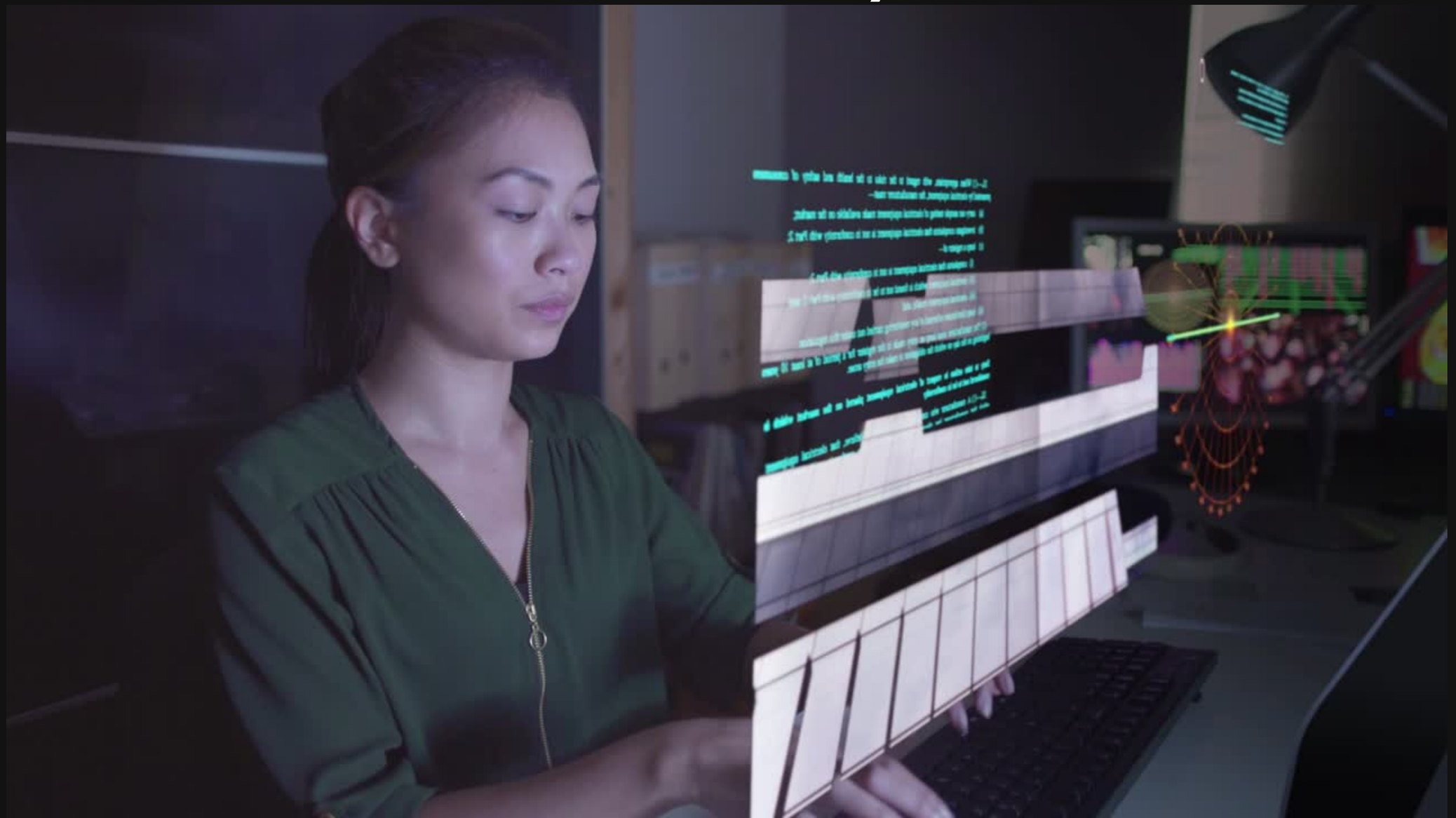
UNIVERSIDADE DE VASSOURAS



Prof. Me. Caio Jannuzzi

VETORES NÃO ORDENADOS

PODEMOS USAR PARA CONTROLAR QUAIS JOGADORES ESTÃO PRESENTES NO CAMPO DE TREINO (ESTRUTURA DE DADOS)



VÁRIAS AÇÕES PODERIAM SER EXECUTADAS, TAIS COMO:

- **INSERIR** UM JOGADOR NA ESTRUTURA DE DADOS QUANDO ELE CHEGAR AO CAMPO DE TREINO;
- **VERIFICAR** SE UM DETERMINADO JOGADOR ESTÁ PRESENTE, PESQUISANDO O NÚMERO DO JOGADOR NA ESTRUTURA
- **REMOVER** UM JOGADOR DA ESTRUTURA DE DADOS QUANDO ELE FOR PARA CASA

INSERÇÃO DE UM NOVO JOGADOR DENTRO DO VETOR

- **Único passo** (inserido na primeira célula vaga do vetor)
- O algoritmo já conhece essa localização porque ele já sabe quantos itens já estão no vetor
- O novo item é simplesmente inserido no próximo espaço disponível
- Big-O constante – **O(1)**

VETORES NÃO ORDENADOS – PESQUISA LINEAR

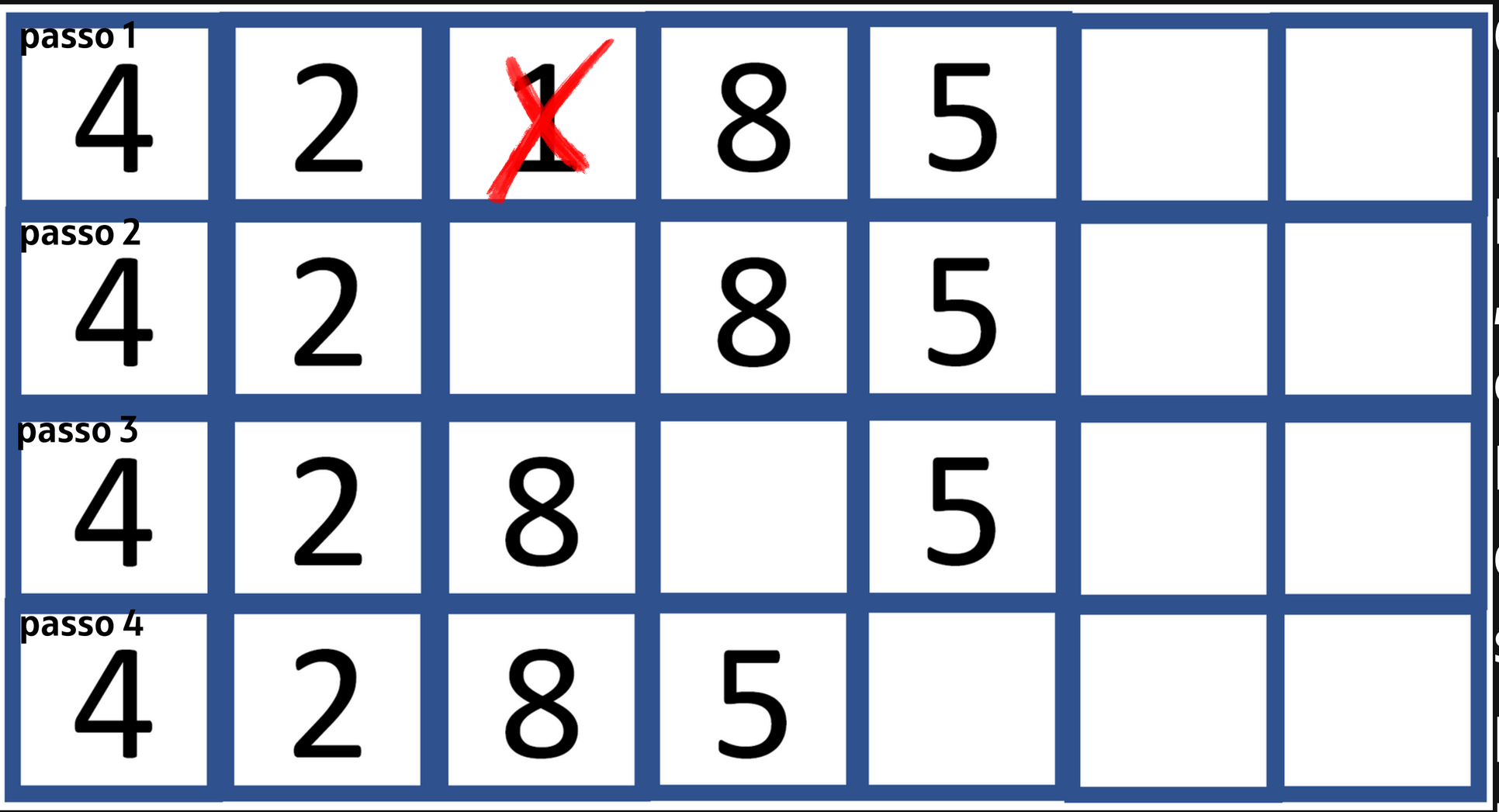
- Percorre cada posição do vetor
- Melhor caso: 4
- Pior caso: 5 ou número que não existe
- Em média, metade dos itens devem ser examinados (N/2)
- Big-O linear – O(n)

VETORES NÃO ORDENADOS – EXCLUSÃO

passo 1	4	2	1	8	5	
passo 2	4	2		8	5	
passo 3	4	2	8		5	
passo 4	4	2	8	5		

Vamos supor que queiramos passar o cerol no número 1 que está no meio do vetor.

VETORES NÃO ORDENADOS – EXCLUSÃO



O número 1 no passo 2 só sumiu por pura didática, deixando o espaço vazio.

Na real a posição dele é sobrescrita pelo próximo item (8 no caso) e assim por diante.

VETORES NÃO ORDENADOS – EXCLUSÃO

- Pesquisar uma média de N/2 elementos (pesquisa linear)
- Pior caso: N
- Mover os elementos restantes (N/2 passos)
- Pior caso: N
- Big-O – O(2n) = O(n)

VETORES NÃO ORDENADOS – DUPLICATAS

- Deve-se decidir se itens com chaves duplicadas serão permitidos
- Exemplo de um arquivo de funcionários
  - Se a chave for o número de registro
  - Se a chave for o sobrenome
- Pesquisa: mesmo se encontrar o valor, o algoritmo terá que continuar procurando até a última célula (N passos)
- Inserção: verificar cada item antes de fazer uma inserção (N passos)
- Exclusão do primeiro item: N/2 comparações e N/2 movimentos
- Exclusão de mais itens: verificar N células e mais de N/2 células

Um exemplo prático.

Vetor não ordenado

+ Code + Markdown

```python
import numpy as np
```

✓ 0.1s Pytho

```python
class VetorNaoOrdenado:

    def __init__(self, capacidade):  # método construtor
        self.capacidade = capacidade  # atributo
        self.ultima_posicao = -1  # atributo
        self.valores = np.empty(
            self.capacidade, dtype=int
        )  # array vario pelo .empty( self.capacidade será o shape e será criado de acordo com o que o usuário passar como parâmetro). dtype=int = tipo inteiro
```

```python
# O(n)
def imprime(self):  # imprime os valores do vetor (array)
    if self.ultima_posicao == -1:  # se o array estiver vazio
        print("O vetor está vazio")
    else:
        for i in range(
            self.ultima_posicao + 1
        ):  # percorre o array de acordo com a quantidade de
        elementos
            print(i, " - ", self.valores[i])  # imprime o
            valor
```

```python
# O(1) - O(2)
def insere(self, valor):  # insere um valor no array
    if (
        self.ultima_posicao == self.capacidade - 1
    ):  # se o array estiver cheio (ultima_posicao =
    capacidade - 1)
        print("Capacidade máxima atingida")
    else:  # se o array não estiver cheio
        self.ultima_posicao += 1  # incrementa a
        ultima_posicao
        self.valores[self.ultima_posicao] = valor  # adiciona
        o valor
```

```python
# O(n)
def pesquisar(
    self, valor
):  # pesquisa um valor no array. UTILIZAREMOS A PESQUISA LINEAR!
    for i in range(self.ultima_posicao + 1):  # percorre o array
        if valor == self.valores[i]:  # se o valor for igual ao valor do array
            return i  # retorna a posicao
    return -1  # se o valor não for encontrado
```

```python
# O(n)
def excluir(self, valor):  # exclui um valor no array
    posicao = self.pesquisar(valor)
    if posicao == -1:  # se o valor não for encontrado
        return -1  #
    else:  # se o valor for encontrado
        for i in range(
            posicao, self.ultima_posicao
        ):  # percorre o array de acordo com a quantidade de
        elementos
            self.valores[i] = self.valores[i + 1]  #
            substitui o valor

        self.ultima_posicao -= 1  # decrementa a
        ultima_posicao
```

Python

```
1  vetor = VetorNaoOrdenado(5)
```

✓ 0.0s

```
1  vetor.imprime()
```

✓ 0.0s

```
O vetor está vazio
```

```
1  vetor.insere(2)
```

✓ 0.0s

```
1  vetor.imprime()
```

✓ 0.0s

```
0  -  2
```

```
1  vetor.insere(3)
2  vetor.insere(5)
3  vetor.insere(8)
4  vetor.insere(1)
```

[7] ✓ 0.0s

```
1  vetor.imprime()
```

[8] ✓ 0.0s

```
0  -  2
1  -  3
2  -  5
3  -  8
4  -  1
```

```
1  vetor.insere(7)
```

[9] ✓ 0.0s

```
Capacidade máxima atingida
```

```
1  vetor.imprime()
```

[10] ✓ 0.0s

```
0  -  2
1  -  3
2  -  5
3  -  8
4  -  1
```

```
1  vetor.pesquisar(8)
```

1] ✓ 0.0s Python

```
3
```

```
1  vetor.pesquisar(9) # valor não encontrado... o que é o pior caso juntamente com o número a ser achado na última posição.
```

1] ✓ 0.0s Python

```
-1
```

```
1  vetor.pesquisar(3)
```

3] ✓ 0.0s Python

```
1
```

```
1  vetor.ultima_posicao
```

[14] ✓ 0.0s

```
4
```

```
1  vetor.imprime()
```

[15] ✓ 0.0s

```
0  -  2
1  -  3
2  -  5
3  -  8
4  -  1
```

```
1  vetor.excluir(5)
```

6] ✓ 0.0s

```
1  vetor.imprime()
```

7] ✓ 0.0s

```
0  -  2
1  -  3
2  -  8
3  -  1
```

```
1  vetor.excluir(1)
2  vetor.imprime()
```

8] ✓ 0.0s

```
0  -  2
1  -  3
2  -  8
```

```
vetor.excluir(2)
vetor.imprime()
```

[19] ✓ 0.0s

```
0 - 3
1 - 8
```

```
vetor.insere(5)
vetor.insere(1)
vetor.imprime()
```

[20] ✓ 0.0s

```
0 - 3
1 - 8
2 - 5
3 - 1
```

Quiz time!

Considere o vetor não ordenado abaixo (sem levar em consideração duplicatas)

15	34	08	12	32	67	43

Quantos passo são necessários para inserir o número **11**?

- ○ **8**
- ○ **9**
- ○ **1**
- ○ **10**

Considere o vetor não ordenado abaixo (sem levar em consideração duplicatas)

15	34	08	12	32	67	43

Quantos passo são necessários para inserir o número **11**?

8

9

1

Somente um passo, o número é inserido no final do vetor

10

Se o vetor abaixo não permitir duplicatas, quantos passos são necessários para inserir o número **33**?

15	34	08	12	32	67	43

- ○ **8**
- ○ **1**
- ○ **5**
- ○ **10**

Se o vetor abaixo não permitir duplicatas, quantos passos são necessários para inserir o número **33**?

15	34	08	12	32	67	43

- (●) 8
- ( ) 1
- ( ) 5
- ( ) 10

7 passos para pesquisar se já existe o 33 e 1 passo para inserir o valor no final do vetor.

Quantos passos são necessários para pesquisar o número **32**?

15	34	08	12	32	67	43

- ○ **1**
- ○ **7**
- ○ **8**
- ○ **5**

Quantos passos são necessários para pesquisar o número **32**?

15	34	08	12	32	67	43

- 1
- 7
- 8

5 passos, comparando todos os número com 32 até chegar no valor procurado

- **5**

Quantos passos são necessários para pesquisar o número **51**?

15	34	08	12	32	67	43

- ○ **7**
- ○ **1**
- ○ **9**
- ○ **0**

Quantos passos são necessários para pesquisar o número **51**?

15	34	08	12	32	67	43

- (●) **7**
- ( ) 1
- ( ) 9
- ( ) 0

7 passos para percorrer todo o vetor e não encontrar o número 51

Se o vetor abaixo permitir duplicatas, quantos passos são necessários para pesquisar o número **12**?

15	34	08	12	32	67	43

- ○ **4**
- ○ **5**
- ○ **1**
- ○ **7**

Se o vetor abaixo permitir duplicatas, quantos passos são necessários para pesquisar o número **12**?

15	34	08	12	32	67	43

4

5

1

7 passos para percorrer todo o vetor, pois mesmo depois que encontrar o número 12, o algoritmo continuará a busca até o final do vetor (pode existir mais números 12)

7

Quantos passos são necessários para remover o número **12**?

15	34	08	12	32	67	43

- ○ **1**
- ○ **7**
- ○ **3**
- ○ **10**

Quantos passos são necessários para remover o número **12**?

15	34	08	12	32	67	43

1

7

3

4 passos para encontrar o número 12, 1 passo para deslocar o 32, 1 passo para deslocar o 67 e 1 passo para deslocar o 43

10

Quantos passos são necessários para remover o número **70** do vetor?

15	34	08	12	32	67	43

- ○ **0**
- ○ **1**
- ○ **7**
- ○ **3**

Quantos passos são necessários para remover o número **70** do vetor?

15	34	08	12	32	67	43

0

1

◉ **7**

3

7 passos para pesquisar o número 70 e chegar no final do vetor e verificar que este número não existe

Inserir um item em um vetor não ordenado que permite duplicatas

- **Leva um tempo proporcional ao tamanho do vetor**
- **Requer diversas comparações**
- **Requer deslocar outros itens para criar espaço**
- **Leva o mesmo tempo não importando quantos itens existem**

Inserir um item em um vetor não ordenado que permite duplicatas

- ○ Leva um tempo proporcional ao tamanho do vetor
- ○ Requer diversas comparações
- ○ Requer deslocar outros itens para criar espaço
- ◉ **Leva o mesmo tempo não importando quantos itens existem**

Se permitir duplicatas terá que percorrer o vetor inteiro para verificar se o elemento já existe


That's all Folks!
ATÉ A PRÓXIMA!