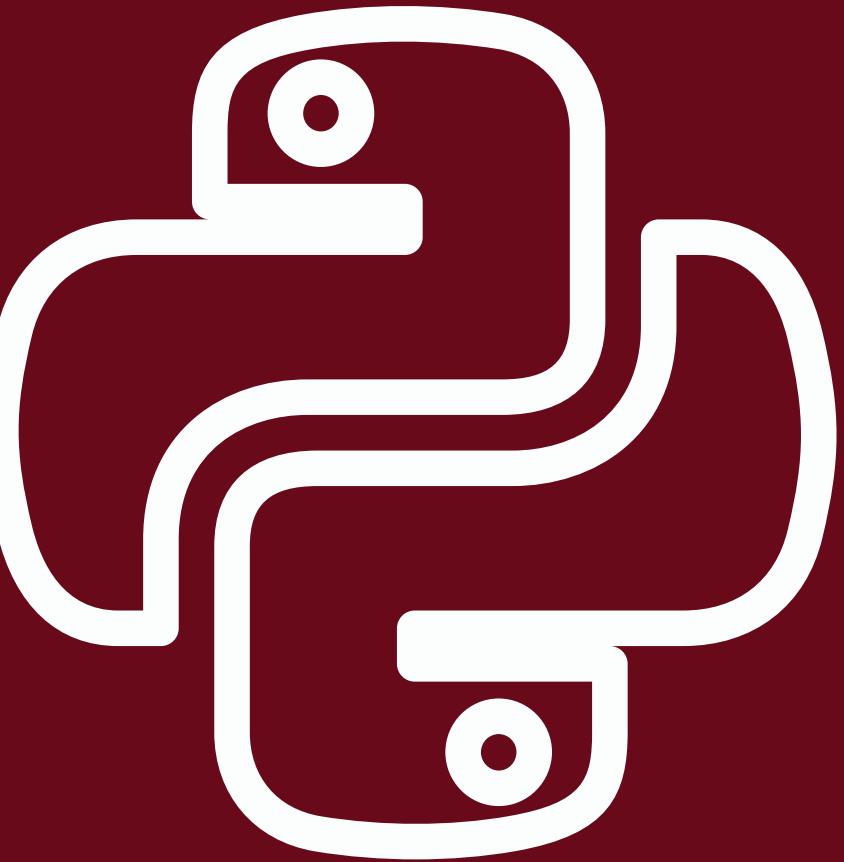


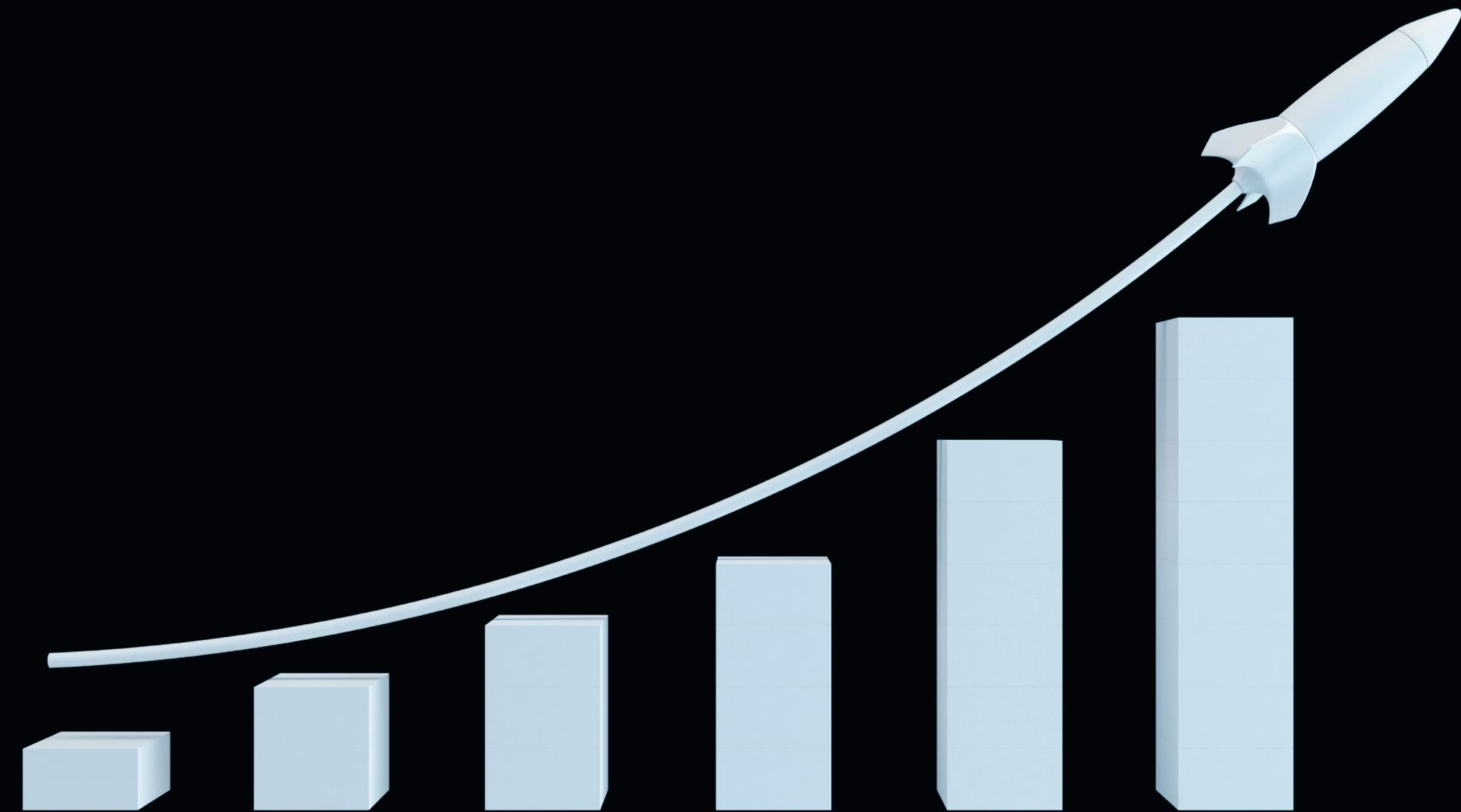
ESTRUTURA DE DADOS



VETORES ORDENADOS



IDEIA MUITO PARECIDA COM O QUE VIMOS NA AULA ANTERIOR, COM O DIFERENCIAL DE QUE DESTA VEZ OS DADOS ESTÃO ORDENADOS!



OS DADOS ESTÃO ORGANIZADOS NA ORDEM ASCENDENTE DE VALORES-CHAVE, OU SEJA, COM O MENOR VALOR NO ÍNDICE 0 E CADA CÉLULA MANTENDO UM VALOR MAIOR QUE A CÉLULA ABAIXO

VANTAGEM: AGILIZA OS TEMPOS DE PESQUISA

1	2	4	5	8		
---	---	---	---	---	--	--

DESVANTAGEM: REQUER MAIS PASSOS PARA A INSERÇÃO... DIFERENTE DO NÃO ORDENADO QUE SIMPLÉSMENTE INSERIA NA ÚLTIMA POSIÇÃO (1 PASSO SOMENTE)

VETORES ORDENADOS - INSERÇÃO

3	1	2	4	5			
1	2	4		5			
1	2		4		5		
1	2			4	5		
1	2	3	4	5			

VETORES ORDENADOS - INSERÇÃO

- 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 ORDENADOS - PESQUISA LINEAR

- A PESQUISA TERMINA QUANDO O PRIMEIRO ITEM MAIOR QUE O VALOR DE PESQUISA É ATINGIDO
- COMO O VETOR ESTÁ ORDENADO, O ALGORITMO SABE QUE NÃO HÁ NECESSIDADE DE PROCURAR MAIS
- **PIOR CASO:** SE O ELEMENTO NÃO ESTIVER NO VETOR OU NA ÚLTIMA POSIÇÃO
- BIG-O - $O(N)$

VETORES ORDENADOS - EXCLUSÃO

1	2	4	5	8		
1	2		5	8		
1	2	5		8		
1	2	5	8			

PARA APAGAR O 4:

1. EXECUTAR PESQUISA LINEAR
2. FAZER O REMANEJAMENTO
3. CADA NÚMERO POSTERIOR VAI OCUPANDO A POSIÇÃO ANTES VAGA

OBSERVAÇÃO

COMO ENVOLVE 2 PASSOS, CASO O ALGORITMO NÃO ENCONTRE NA LISTA O NÚMERO PARA EXCLUSÃO ELE TERMINA LOGO NO PASSO 1 QUE É A PESQUISA.

VETORES ORDENADOS - EXCLUSÃO

- O ALGORITMO PODE TERMINAR NA METADE DO CAMINHO SE NÃO ENCONTRAR O ITEM.
- 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 ORDENADOS - EXCLUSÃO

- O ALGORITMO PODE TERMINAR NA METADE DO CAMINHO SE NÃO ENCONTRAR O ITEM.
- PESQUISAR UMA MÉDIA DE $N/2$ ELEMENTOS (PESQUISA LINEAR)
 - PIOR CASO: N
- MOVER OS ELEMENTOS RESTANTES ($N/2$ PASSOS)
 - PIOR CASO: N
- BIG-O - $O(2N) = O(N)$

Implementação

Inserção

Vetor ordenado

```
1 import numpy as np  
[✓] 0.0s
```

```
1 class VetorOrdenado:  
2  
3     def __init__(  
4         self, capacidade  
5     ): # método construtor que recebe como parametro a capacidade do vetor.  
6         self.capacidade = capacidade # capacidade do vetor  
7         self.ultima_posicao = -1 # indica que o vetor esta vazio  
8         self.valores = np.empty(  
9             self.capacidade, dtype=int  
10        ) # cria um array numpy vazio com capacidade igual a capacidade do vetor  
11  
12    # O(n)  
13    def imprime(self): # imprime o vetor  
14        if self.ultima_posicao == -1: # se o vetor estiver vazio  
15            print("O vetor está vazio")  
16        else:  
17            for i in range(self.ultima_posicao + 1): # percorre o vetor  
18                print(i, " - ", self.valores[i]) # imprime a posicao e o valor
```

```
def insere(self, valor): # insere um valor no vetor ao passar o parametro valor
    if (
        self.ultima_posicao == self.capacidade - 1
    ): # se a ultima posicao for igual a capacidade do vetor - 1
        print("Capacidade máxima atingida")
        return

    posicao = 0 # posicao recebe 0
    for i in range(self.ultima_posicao + 1): # percorre todos os elementos do vetor
        posicao = i # posicao recebe i (recebe a posicao do vetor a medida que for avançando na estrutura de dados)
        if (
            self.valores[i] > valor
        ): # se o valor na posicao i for maior que o valor passado como parametro
            break # sai do loop quando o valor na posicao i for maior que o valor passado como parametro
        if i == self.ultima_posicao: # se i for igual a ultima posicao
            posicao = i + 1 # posicao recebe i + 1

    x = self.ultima_posicao # x recebe a ultima posicao
    while x ≥ posicao: # enquanto x for maior ou igual a posicao
        self.valores[x + 1] = self.valores[
            x
        ] # o valor na posicao x + 1 recebe o valor na posicao x
        x -= 1 # x recebe x - 1

    self.valores[posicao] = (
        valor # o valor na posicao recebe o valor passado como parametro
    )
    self.ultima_posicao += 1 # a ultima posicao recebe a ultima posicao + 1
```

```
1 vetor = VetorOrdenado(10) # cria um objeto do tipo VetorOrdenado com capacidade 10
2 vetor.imprime()
```

✓ 0.0s

0 vetor está vazio

```
1 vetor.insere(6) # insere o valor 6 no vetor
2 vetor.imprime() # imprime o vetor
```

✓ 0.0s

0 - 6

```
1 vetor.insere(4) # insere o valor 4 no vetor
2 vetor.imprime() # imprime o vetor
+] ✓ 0.0s

0 - 4
1 - 6
```

```
1 vetor.insere(3) # insere o valor 3 no vetor
2 vetor.imprime() # imprime o vetor
+] ✓ 0.0s

0 - 3
1 - 4
2 - 6
```

```
1 vetor.insere(5) # insere o valor 5 no vetor
2 vetor.imprime() # imprime o vetor
+] ✓ 0.0s

0 - 3
1 - 4
2 - 5
3 - 6
```

```
> <...>
    1 vetor.insere(1) # insere o valor 1 no vetor
    2 vetor.imprime() # imprime o vetor
17] ✓ 0.0s
```

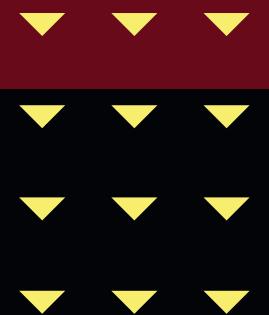
```
.. 0 - 1
  1 - 3
  2 - 4
  3 - 5
  4 - 6
```

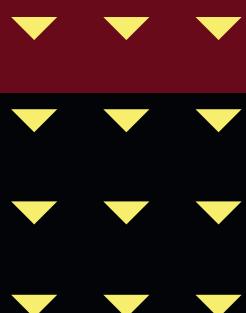
```
1 vetor.insere(8) # insere o valor 8 no vetor
2 vetor.imprime() # imprime o vetor
18] ✓ 0.0s
```

```
.. 0 - 1
  1 - 3
  2 - 4
  3 - 5
  4 - 6
  5 - 8
```

Implementação

Pesquisa





Adicionem o método pesquisa a classe **VetorOrdenado**

```
# O(n)
def pesquisar(self, valor): # pesquisa um valor no vetor ao
passar o parametro valor
    for i in range(self.ultima_posicao + 1): # percorre o
vetor
        if self.valores[i] > valor: # se o valor na posicao i
for maior que o valor passado como parametro
            return -1
        if self.valores[i] == valor: # se o valor na posicao
i for igual ao valor passado como parametro
            return i
        if i == self.ultima_posicao: # se i for igual a
ultima posicao
            return -1
```

1 vetor.pesquisar(5) # pesquisa o valor 5 no vetor. levou 4 passos
para encontrar o valor 5. $O(n)$

2

✓ 0.0s

Python

3

1 vetor.pesquisar(8) # pesquisa o valor 8 no vetor. levou 6 passos
para encontrar o valor 8. $O(n)$

✓ 0.0s

Python

5

```
1 vetor.pesquisar(2) # pesquisa o valor 2 no vetor. levou 2 passos  
para percorrer o vetor pesquisando o valor 2 que não existe pois  
ele já teria passado para o número 3 e não achou o número 2 antes  
dele. O(n). O return -1 indica que o valor não foi encontrado.
```

✓ 0.0s

Python

-1



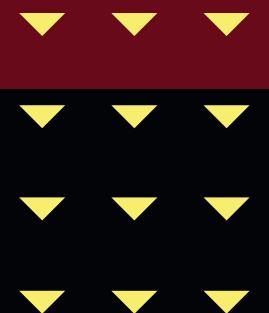
```
1 vetor.pesquisar(9) # pesquisa o valor 9 no vetor. levou 6 passos  
para percorrer o vetor pesquisando o valor 9 que não existe pois  
ele já teria passado o número 8 que é o último do array. O(n). O  
return -1 indica que o valor não foi encontrado.
```

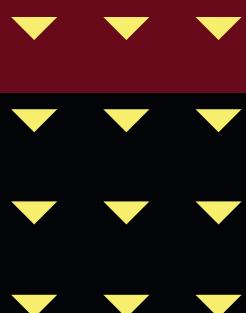
✓ 0.0s

Python

-1

Implementação exclusão





Adicionem o **método excluir** a classe **VetorOrdenado**

```
def excluir(self, valor): # exclui um valor no vetor ao  
passar o parametro valor  
    posicao= self.pesquisar(valor) # posicao recebe o valor  
    retornado pelo metodo pesquisar  
    if posicao == -1:# se a posicao for igual a -1  
        return -1  
    else:  
        for i in range(posicao, self.ultima_posicao): #  
            percorre o vetor a partir da posicao ate a ultima  
            posicao  
            self.valores[i] = self.valores[i+1] # o valor na  
            posicao i recebe o valor na posicao i + 1  
            self.ultima_posicao -= 1 # a ultima posicao recebe a  
            ultima posicao - 1
```

```
1 vetor.imprime() # imprime o vetor
```

✓ 0.0s

Python

0	-	1
1	-	3
2	-	4
3	-	5
4	-	6
5	-	8

```
1 vetor.excluir(5) # exclui o valor 5 do vetor
2 vetor.imprime() # imprime o vetor
```

✓ 0.0s

Python

0	-	1
1	-	3
2	-	4
3	-	6
4	-	8

```
1 vetor.excluir(1) # exclui o valor 1 do vetor  
2 vetor.imprime() # imprime o vetor
```

] ✓ 0.0s

Pyt

```
0 - 3  
1 - 4  
2 - 6  
3 - 8
```

```
1 vetor.excluir(8) # exclui o valor 8 do vetor  
2 vetor.imprime() # imprime o vetor
```

] ✓ 0.0s

Pyt

```
0 - 3  
1 - 4  
2 - 6
```



```
1 vetor.excluir(9) # tenta excluir o valor 9 que não existe no  
vetor
```

2

] ✓ 0.0s

Pyt

-1

Vetores Ordenados

PESQUISA BINÁRIA

Vamos supor que tenhamos números de 1 até 100
(dentro de um **vetor** por exemplo)

Queremos pesquisar/adivinar o número 47.

Caso fosse uma **PESQUISA LINEAR**... seriam 47 passos.. do 1 ao 47 seguindo a ordem do vetor.

Porém temos uma maneira mais fácil!

Vamos utilizar a pesquisa binária.

Vamos dividir por 2 nosso problema...

de 1 até $100/2 = 50$.

50 é o número pesquisado? Não!

47 é menor ou maior do que 50? Menor!

Vamos utilizar a pesquisa binária.

Vamos dividir por 2 nosso problema...

de 1 até $100/2 = 50$.

50 é o número pesquisado? Não!

47 é menor ou maior do que 50? Menor! Logo, não faz sentido fazer da posição 50 até a 100 a pesquisa.

Agora sabendo que é menor que 50... nossa base de dados fica menor e melhor de se trabalhar!

$$1 \text{ até } 49 / 2 = 25$$

25 é o número pesquisado? **Não!**

47 é menor ou maior do que 25? **Maior!**

Agora sabendo que é menor que 50 e maior que 25
o nosso número que queremos adivinhar... nossa
base de dados fica menor e melhor de se trabalhar!

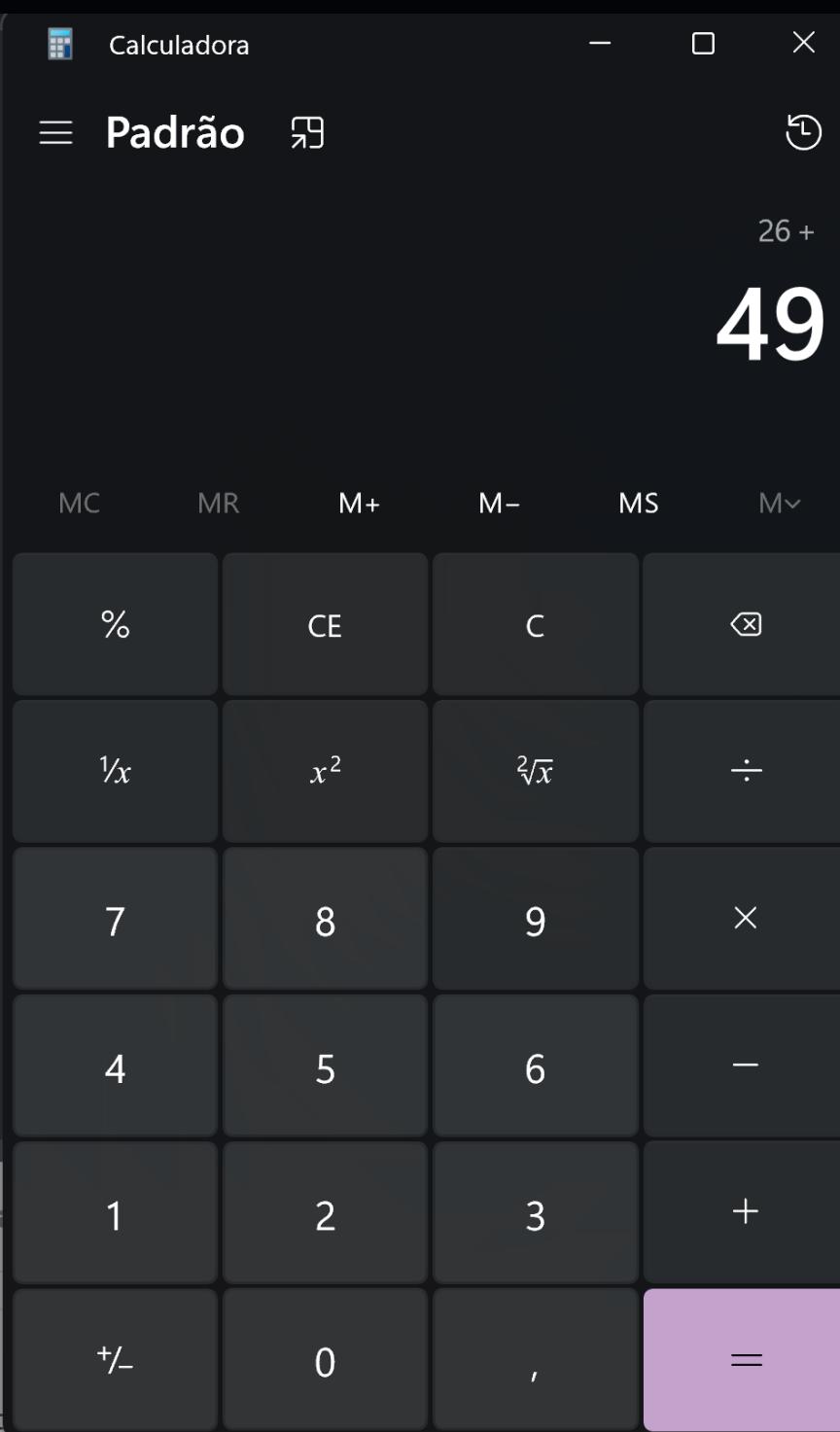
$$26 \text{ até } 49 / 2 = 38$$

38 é o número pesquisado? **Não!**

47 é menor ou maior do que 38? **Maior!**

“Caio, estou nerfado em matemática hoje também”

De onde saiu o 38 do slide anterior?



Sim!
Aproximei
para 38

Agora sabendo que é menor que 50 e maior que 38
o nosso número que queremos adivinhar... nossa
base de dados fica menor e melhor de se trabalhar!

$$39 \text{ até } 49 / 2 = 44$$

44 é o número pesquisado? **Não!**

47 é menor ou maior do que 44? **Maior!**

Agora sabendo que é menor que 50 e maior que 44
o nosso número que queremos adivinhar... nossa
base de dados fica menor e melhor de se trabalhar!

$$45 \text{ até } 49 / 2 = 47$$

47 é o número pesquisado? Sim

Quantos passos executamos para encontrar o número?

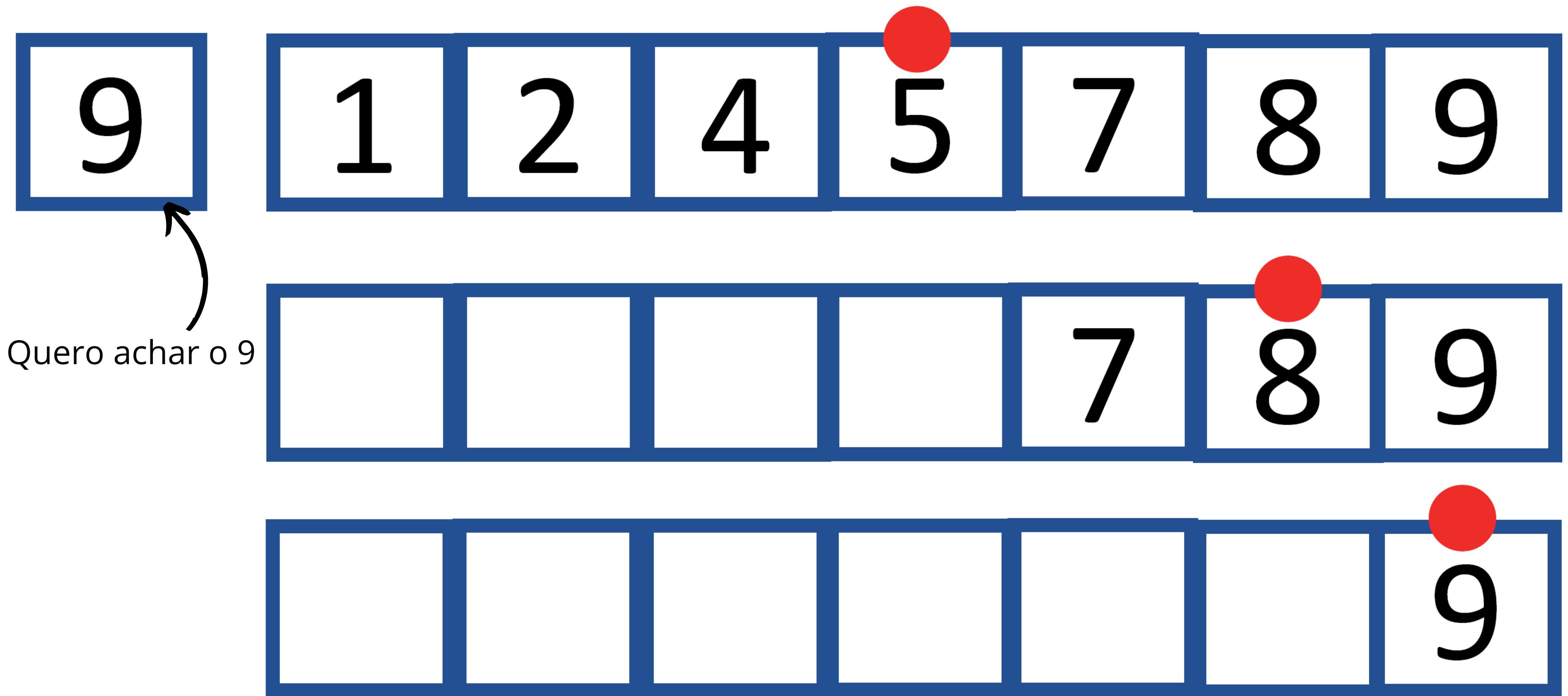
- Números de 1 até 100
 - Pesquisar/adivinar o número 47
-
- 1 • $1 \text{ até } 100 / 2 = 50$
 - 50 é o número pesquisado? Não
 - 47 é menor ou maior do que 50? Menor
 - 2 • $1 \text{ até } 49 / 2 = 25$
 - 25 é o número pesquisado? Não
 - 47 é menor ou maior do que 25? Maior
 - 3 • $26 \text{ até } 49 / 2 = 38$
 - 38 é o número pesquisado? Não
 - 47 é menor ou maior do que 38? Maior
 - 4 • $39 \text{ até } 49 / 2 = 44$
 - 44 é o número pesquisado? Não
 - 47 é menor ou maior do que 44? Maior
 - 5 • $45 \text{ até } 49 / 2 = 47$
 - 47 é o número pesquisado? Sim

Somente 5
passos!





PESQUISA BINÁRIA EXEMPLO UTILIZANDO UM VETOR ORDENADO



9

3 NÚMEROS PARA LÁ

1 2 4 5 7 8 9

3 NÚMEROS PARA CÁ

DIVIDO NO
MEIO O VETOR

7 8 9

9



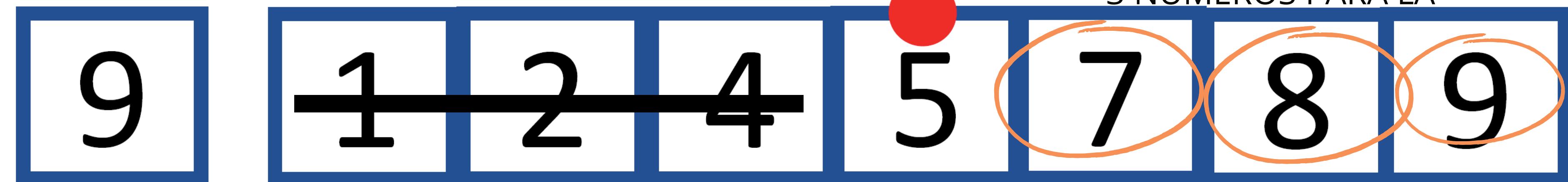
O 5 É O NÚMERO PESQUISADO?

NÃO!

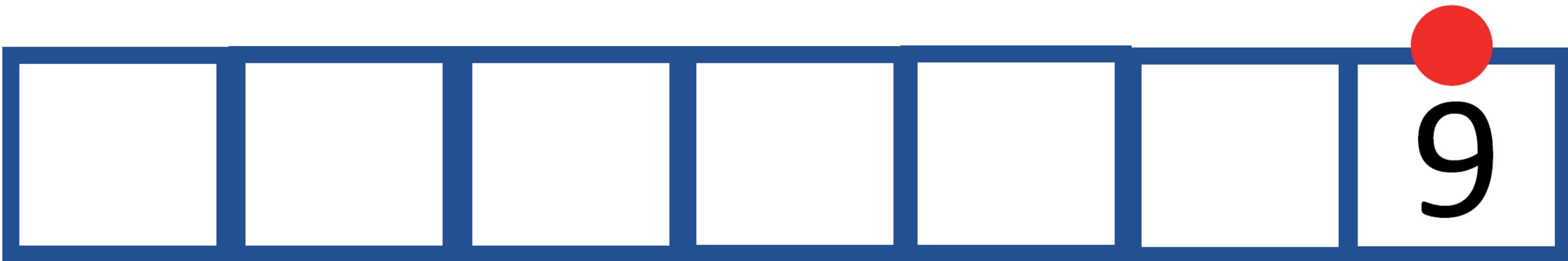
9 É MAIOR OU MENOR QUE 5?

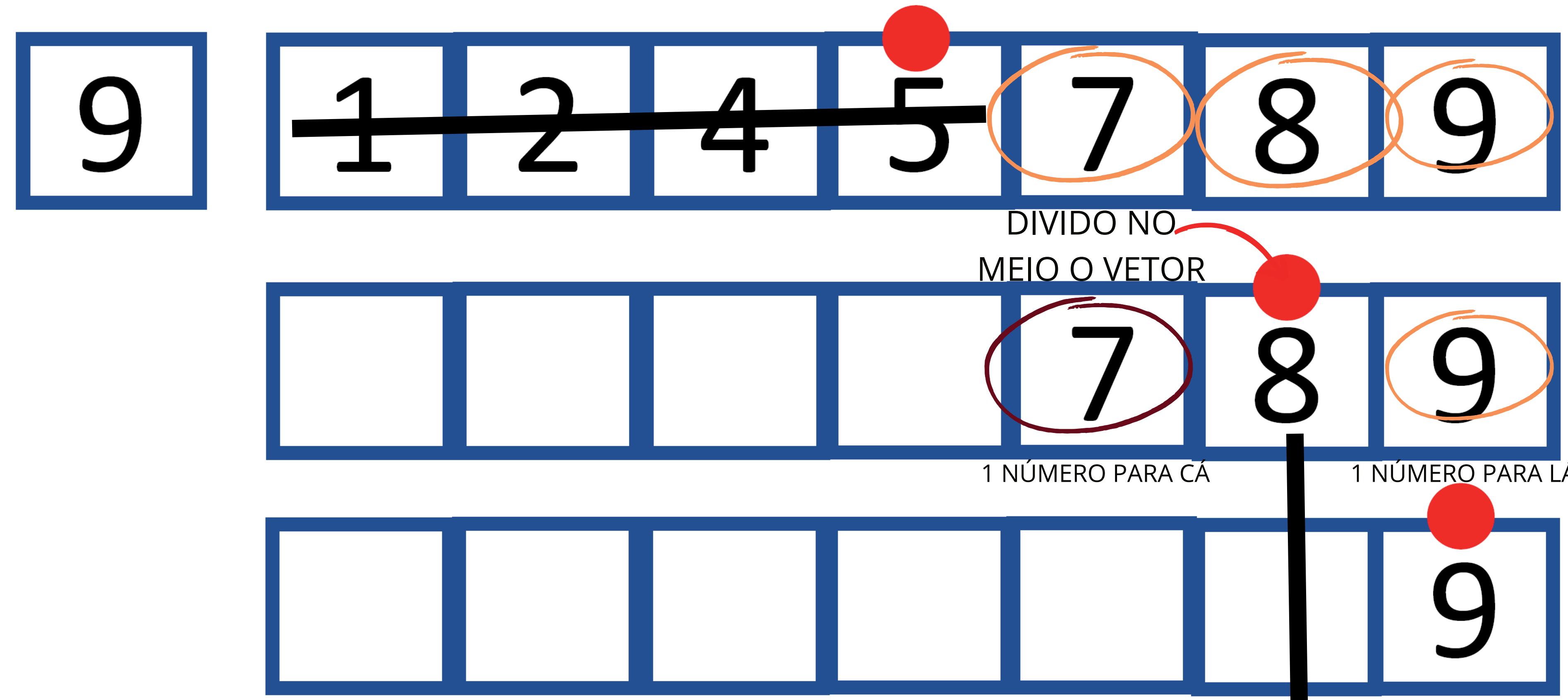
MAIOR!

3 NÚMEROS PARA LÁ



Como o 9 é maior que 5 irei buscar somente na parte da direita







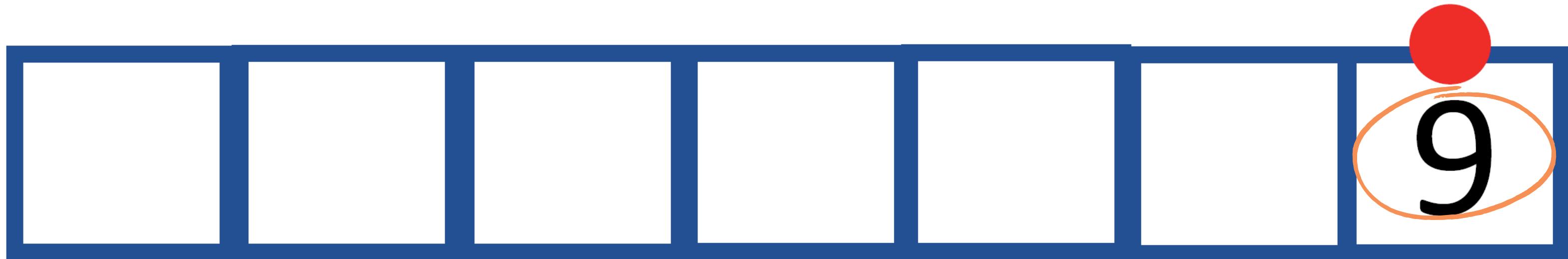
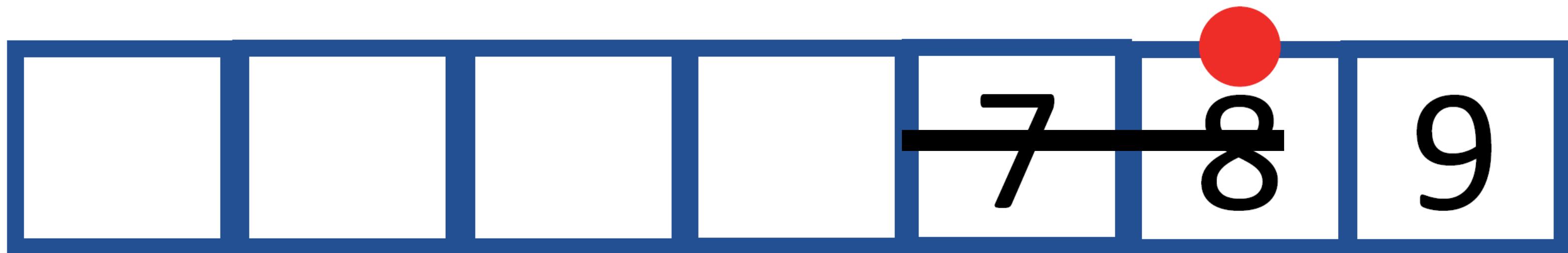
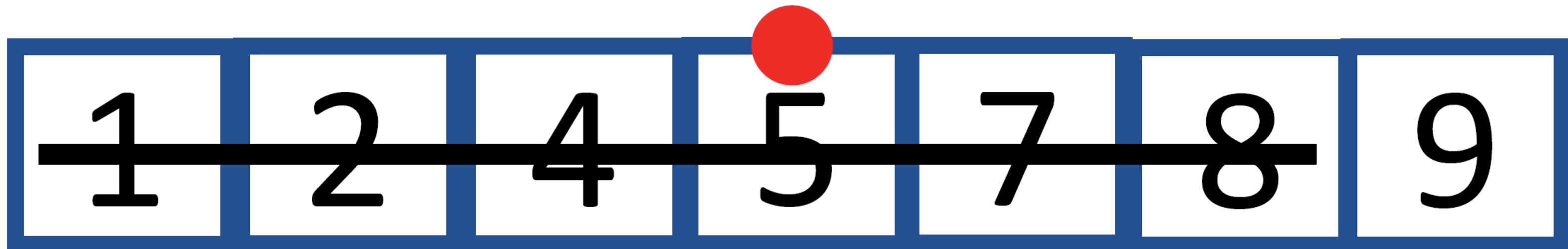
O 8 É O NÚMERO PESQUISADO?

NÃO!

9 É MAIOR OU MENOR QUE 8?

MAIOR!

9



PESQUISA BINÁRIA x PESQUISA LINEAR

Faixa (valores do array)	Comparações Binária (número de passos até encontrar)	Comparações Linear (N/2) número de passos até encontrar
10	4	5
100	7	50
1.000	10	500
10.000	14	5.000
100.000	17	50.000
1.000.000	20	500.000
10.000.000	24	5.000.000
100.000.000	27	50.000.000
1.000.000.000	30	500.000.000

Implementação pesquisa binária





Adicionem o método `pesquisa_binaria` a classe
`VetorOrdenado`

sem zoom

```
# O(log n)
def pesquisa_binaria(self, valor): # pesquisa um valor no vetor ao passar o parametro valor
    limite_inferior = 0 # limite inferior recebe 0
    limite_superior = self.ultima_posicao # limite superior recebe a ultima posicao

    while True: # loop infinito enquanto True
        posicao_atual = int((limite_inferior + limite_superior) / 2) # posicao atual recebe a media entre o limite inferior e o limite superior
        # Se achou na primeira tentativa
        if self.valores[posicao_atual] == valor: # se o valor na posicao atual for igual ao valor passado como parametro
            return posicao_atual
        # Se não achou
        elif limite_inferior > limite_superior: # se o limite inferior for maior que o limite superior
            return -1
        # Divide os limites
        else:
            # Limite inferior
            if self.valores[posicao_atual] < valor:# se o valor na posicao atual for menor que o valor passado como parametro
                limite_inferior = posicao_atual + 1
            # Limite superior
            else: # se o valor na posicao atual for maior que o valor passado como parametro
                limite_superior = posicao_atual - 1
```

com zoom

```
# O(log n)
def pesquisa_binaria(self, valor): # pesquisa um valor no vetor ao passo
    limite_inferior = 0 # limite inferior recebe 0
    limite_superior = self.ultima_posicao # limite superior recebe a ultima posicao
    while True: # loop infinito enquanto True
        posicao_atual = int((limite_inferior + limite_superior) / 2)
        # Se achou na primeira tentativa
        if self.valores[posicao_atual] == valor: # se o valor na posicao atual for igual ao valor procurado
            return posicao_atual
        # Se não achou
        elif limite_inferior > limite_superior: # se o limite inferior for maior que o limite superior
            return -1
        # Divide os limites
        else:
            # Limite inferior
            if self.valores[posicao_atual] < valor:# se o valor na posicao atual for menor que o valor procurado
                limite_inferior = posicao_atual + 1
            # Limite superior
            else: # se o valor na posicao atual for maior que o valor procurado
                limite_superior = posicao_atual - 1
```

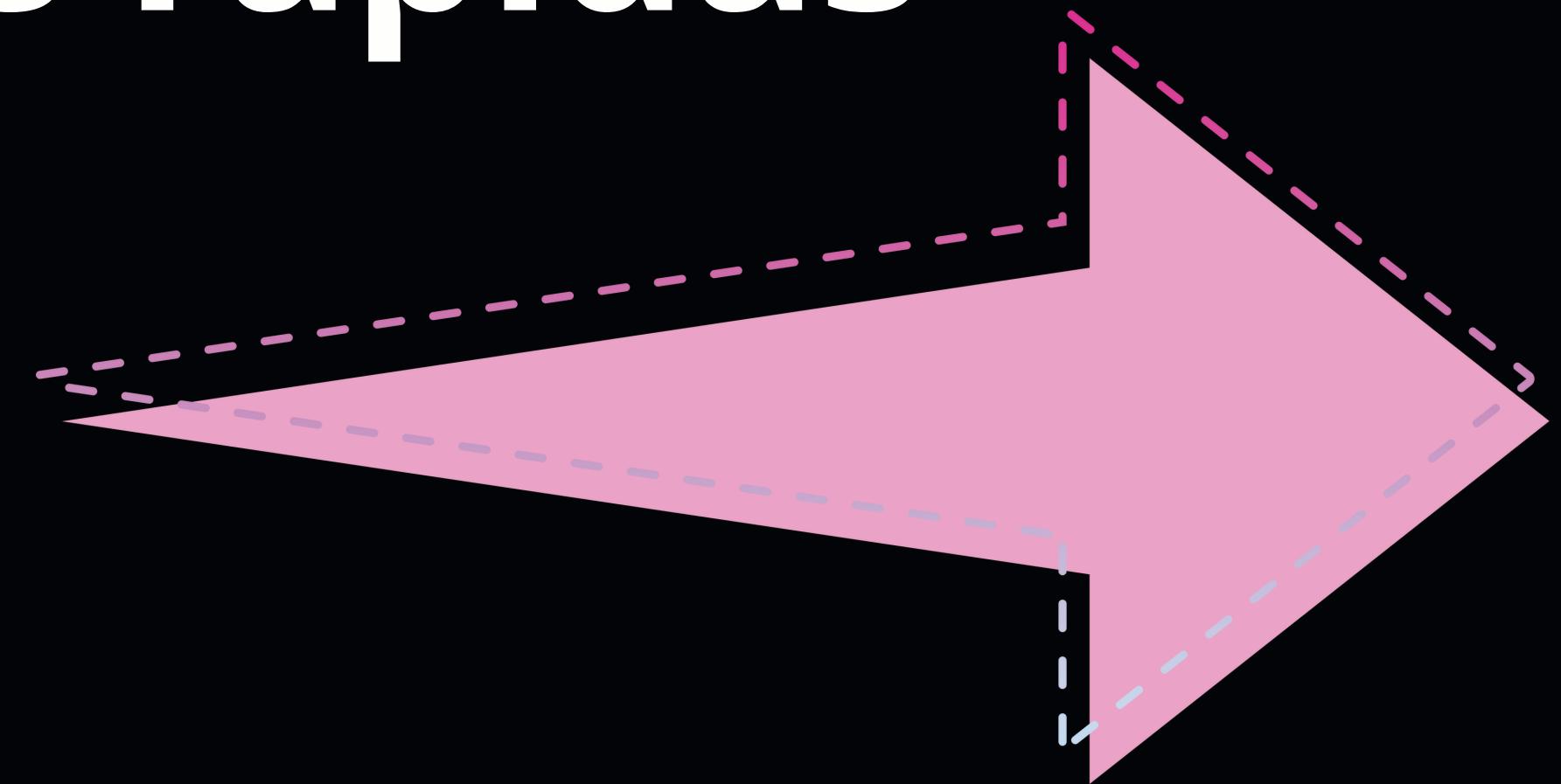
```
1 vetor = VetorOrdenado(10) # cria um objeto do tipo VetorOrdenado  
com capacidade 10  
2 vetor.insere(8) # insere o valor 8 no vetor  
3 vetor.insere(9) # insere o valor 9 no vetor  
4 vetor.insere(4) # insere o valor 4 no vetor  
5 vetor.insere(1) # insere o valor 1 no vetor  
6 vetor.insere(5) # insere o valor 5 no vetor  
7 vetor.insere(7) # insere o valor 7 no vetor  
8 vetor.insere(11) # insere o valor 11 no vetor  
9 vetor.insere(13) # insere o valor 13 no vetor  
10 vetor.insere(2) # insere o valor 2 no vetor  
11 vetor.imprime() # imprime o vetor
```

] ✓ 0.0s

Python

```
0 - 1  
1 - 2  
2 - 4  
3 - 5  
4 - 7  
5 - 8  
6 - 9  
7 - 11  
8 - 13
```


Discussões rápidas



- A principal **vantagem** é que os tempos de pesquisa são muito mais rápidos do que em um vetor não ordenado.
- A **inserção** leva **mais tempo**, pois os itens de dados devem ser movidos para criar espaço.
- As **remoções** são **lentas** tanto nos vetores ordenados quanto nos não ordenados, pois os itens devem ser movidos para preencher o “buraco”.
- Vetores **ordenados** são **úteis** quando pesquisas são **frequentes**, mas inserções e remoções não são.



Quiz time!

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para pesquisar o número 33?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

1

7

4

15

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para pesquisar o número 33?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

- 1
- 7
- 4

4 passos para comparar todos os valores até o 33

- 15

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para pesquisar o número 11?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

1

7

8

2

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para pesquisar o número 11?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

1

7

8

2 passos pois as comparações são feitas até o número 20

2

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para remover o número 33?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

8

15

1

4

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para remover o número 33?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

8

4 passos para pesquisar o local, 3 passos para mover os elementos e 1 passo para inserir o número

15

1

4

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para excluir o número 35?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

10

1

7

5

Considere o vetor ordenado abaixo e utilizando pesquisa linear. Quantos passos são necessários para excluir o número 35?

10	20	24	33	36	44	59
----	----	----	----	----	----	----

10

1

7

5 passos até fazer a pesquisa até o número 35

5

Inserir um item em um vetor não ordenado que permita duplicatas (marque somente a alternativa correta)

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 permita duplicatas (marque somente a alternativa correta)

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

Vetores ordenados, comparados com vetores não ordenados, são (marque somente uma alternativa)

Muito mais rápidos na remoção

Mais rápidos na inserção

Mais rápidos para criar

Mais rápidos na pesquisa

Vetores ordenados, comparados com vetores não ordenados, são (marque somente uma alternativa)

Muito mais rápidos na remoção

Mais rápidos na inserção

Mais rápidos para criar

Mais rápidos na pesquisa

Quando você remove um item de um vetor não ordenado, na maioria dos casos você desloca outros itens para preencher o intervalo

Verdadeiro

Falso

Quando você remove um item de um vetor não ordenado, na maioria dos casos você desloca outros itens para preencher o intervalo

Verdadeiro

Falso

Uma pesquisa binária pode ser aplicada em um vetor não ordenado

Verdadeiro

Falso

Uma pesquisa binária pode ser aplicada em um vetor não ordenado

Verdadeiro

Falso

Pesquisas lineares requerem um tempo proporcional ao número de itens em um vetor

Verdadeiro

Falso

Pesquisas lineares requerem um tempo proporcional ao número de itens em um vetor

Verdadeiro

Falso

That's all Folks!

ATÉ A PRÓXIMA!