

Programação 1

# Busca em vetores

[pcardoso@ufpa.br](mailto:pcardoso@ufpa.br)

# Algoritmos de busca em vetores

- Contextualização

- Muitas vezes, é preciso verificar se um dado elemento está presente em um vetor e em qual posição.
- Em outras palavras, dado um vetor **V** de **N** posições, queremos determinar se um elemento **K** está ou não no vetor.



# Algoritmos de busca em vetores

- Contextualização

- Por exemplo: dado  $\mathbf{V} = \{2, -3, 101, 104, 0, 1, 2, 3, -3, 0\}$  com  $\mathbf{N} = 10$  posições, queremos saber se  $\mathbf{K} = 104$  está presente ou não no vetor. Em caso afirmativo, queremos saber também qual posição ele ocupa.



$\mathbf{K} \leftarrow 104$

Neste exemplo,  $\mathbf{K} \in \mathbf{V}$  e ocupa a posição (índice) **3** do vetor.

# Algoritmos de busca em vetores

- Contextualização
  - O exemplo anterior foi bem simples, agora imagine verificar se um dado elemento está presente em um vetor com milhares ou milhões de elementos. **Como resolver este problema?**
  - Existem na literatura, diversos algoritmos que resolvem o problema da busca em vetores. Nesta disciplina, veremos duas estratégias:
    - **Busca linear ou sequencial.**
    - **Busca binária.**

# Algoritmos de busca em vetores

- **Busca linear ou sequencial**

- Consiste em verificar sequencialmente as posições em um vetor **V** de **N** elementos, uma a uma, até encontrar o elemento **K** ou chegar ao final do vetor.
  - No **melhor** caso, verificamos apenas **1** posição do vetor (o primeiro elemento investigado do vetor é o próprio **K**).
  - No **pior** caso, verificamos **N** posições (**K** é o último elemento investigado do vetor ou ele não está presente no mesmo).
  - No caso **médio**, verificamos **N/2** posições (**K** está em alguma posição mediana do vetor).

# Algoritmos de busca em vetores



- **Busca linear ou sequencial**

- A seguir, no próximo slide, será apresentado um exemplo de busca sequencial em um vetor de números inteiros.
- **Atenção:** o exemplo de implementação do próximo slide é apenas isso, um exemplo. É possível implementar o conceito da busca sequencial de diferentes formas e utilizando vetores de diferentes tipos de dados.
  - **Recomendação:** não tente decorar a implementação do exemplo. Tente entender como a busca sequencial funciona!

# Exemplo: busca sequencial

```
v=[3,70,20,15,88,49]
k=int(input("Informe um valor para buscar: "))
posicao=-1
tam=len(v)
for i in range (tam):
    if v[i]==k:
        posicao=i

print(posicao)
```

## Variáveis

- **k**: elemento a ser buscado no vetor.
- **tam**: quantidade de elementos do vetor.
- **V**: vetor de elementos.
- **i**: variável auxiliar para caminhar nas posições do vetor.
- **posicao**: variável que armazena a posição do elemento **buscar** no vetor **V**, caso ele seja encontrado.

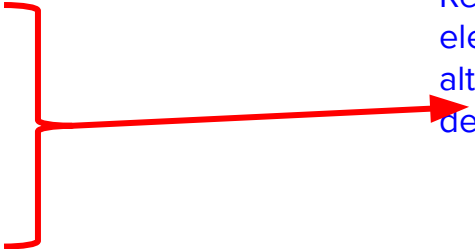
# Exemplo: busca sequencial

```
v=[3,70,20,15,88,49]
k=int(input("Informe um valor para buscar: "))
posicao=-1
tam=len(v)
for i in range (tam):
    if v[i]==k:
        posicao=i

print(posicao)
```

Inicialização da variável **posicao**. Nesta implementação, estamos assumindo o valor **-1** para indicar que o elemento não foi encontrado

Repetição da busca linear. Note que, caso o elemento **k** seja encontrado no vetor **V**, alteramos o valor da variável **posicao** para demarcar o índice no qual ele foi encontrado.





# Algoritmos de busca em vetores

- **Busca linear ou sequencial**

- É possível melhorar a eficiência da busca no algoritmo do exemplo de implementação apresentado.
- Note que, na implementação apresentada, mesmo que encontremos o elemento **K** no vetor **V**, ainda precisamos percorrer todos os índices presentes no vetor até que cheguemos no valor de **N** (pois construímos a repetição *para ( $i < N$ )* ).
  - Sendo assim, podemos melhorar nosso algoritmo, interrompendo a busca tão logo identifiquemos que **K** está presente em **V**.

# Busca sequencial melhorada

```
v=[3,70,20,15,88,49]
k=int(input("Informe um valor para buscar: "))
posicao=0

while posicao < len(v) and v[posicao]!=k:
    posicao=posicao+1

if posicao!=len(v):
    print(posicao)
```

Repetição melhorada da busca linear. Note que neste caso avançamos o valor do índice **posicao** enquanto não encontrarmos o elemento **k** ou chegarmos no final do vetor.

Se ao sair do comando **while**, o valor de **posicao** não é igual a **N** (quantidade de elementos), isto quer dizer que saímos do **while** porque encontramos o elemento dentro do vetor e, portanto, podemos registrar o último índice **i** assumido dentro da repetição como sendo a posição do elemento **K** no vetor **V**.

# Algoritmos de busca em vetores

- **Busca binária**

- É possível melhorar a eficiência da busca em vetores ordenados se nós modificarmos a nossa estratégia do processo de busca.
- A busca binária é um método de busca em vetores ordenados e que reduz o tamanho do problema (quantidade de elementos a serem investigados) a cada iteração.
  - A ideia básica é ir “quebrando” o vetor ao meio, em cada iteração, e verificar em qual metade pode estar o valor. Repete-se o processo até que o vetor de busca tenha tamanho 1, sendo encontrado ou não o elemento de interesse.

# Algoritmos de busca em vetores

- **Busca binária**

- Seja **V** um vetor de **N** posições e queremos determinar se um elemento **K** está ou não presente no vetor, o algoritmo geral em pseudocódigo da

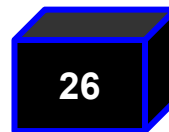
Pseudocódigo da busca binária em vetores de ordem crescente

1. Comparamos **K** com o elemento do meio do vetor ordenado em ordem crescente.
2. Se o elemento do meio for igual a **K**, terminamos a busca, pois encontramos o elemento.
3. Se o elemento do meio for maior do que **K**, a metade da direita pode ser descartada da busca.
4. Se o elemento do meio for menor do que **K**, a metade da esquerda pode ser descartada da busca.
5. Repetimos então este mesmo processo com a metade restante do vetor. Se esta metade for vazia, conclui-se que o elemento não está presente no vetor.

# Algoritmos de busca em vetores

- **Busca binária**

- Exemplo: suponha que desejamos buscar  $\kappa \leftarrow 26$

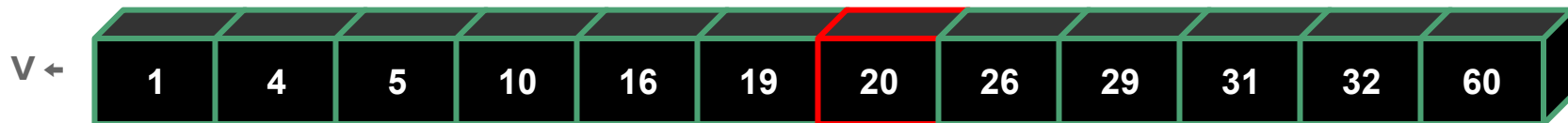
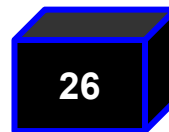


**OBS:** Considere neste exemplo que os índices do vetor **V** vão de 1 a 12

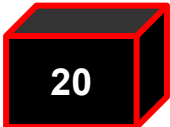
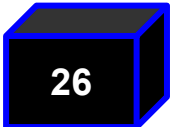
# Algoritmos de busca em vetores

- Busca binária

- Exemplo: suponha que desejamos buscar  $\kappa \leftarrow 26$



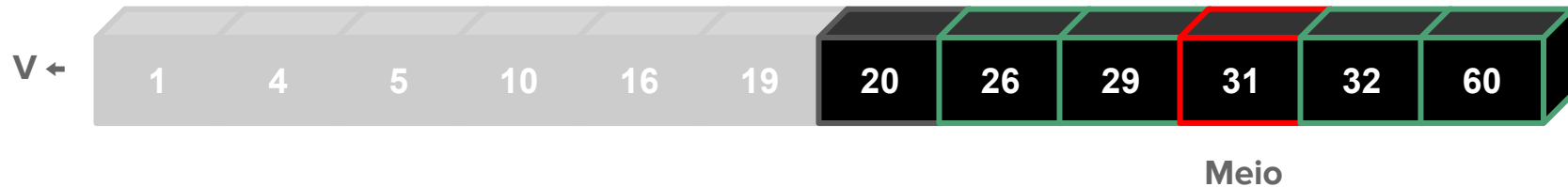
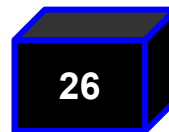
Meio

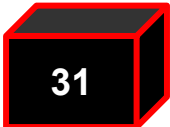
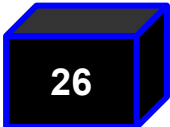
Como   $<$   podemos descartar a metade esquerda do vetor

# Algoritmos de busca em vetores

- **Busca binária**

- Exemplo: suponha que desejamos buscar  $\kappa \leftarrow 26$

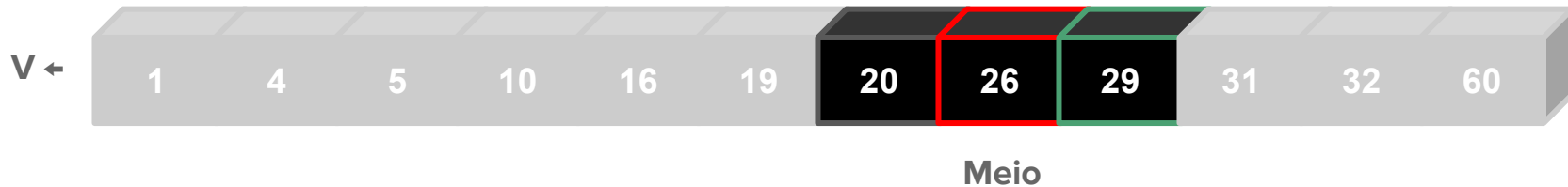
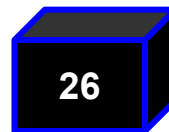


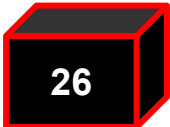
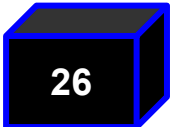
Como   $>$   podemos descartar a metade direita do vetor

# Algoritmos de busca em vetores

- Busca binária

- Exemplo: suponha que desejamos buscar  $\kappa \leftarrow 26$



Como  =  podemos encerrar a busca, pois encontramos o elemento



# Busca binária em lista - iterativa


```
v=[3,15,70,80,150,461]
k=int(input())
inicio = 0
fim = len(v)-1
achou=False
while inicio<=fim and not achou:
    meio = (inicio + fim)//2
    if v[meio] == k:
        achou=True
    else:
        if k < v[meio]:
            fim = meio-1
        else:
            inicio=meio+1
print(achou,meio)
```


← // Resulta em um valor inteiro


← Uso de sentinela para controle

# Busca binária em lista - recursiva

```
def buscaRecurativa(v,item,inicio, fim):  
    if inicio<=fim:  
        meio = (inicio + fim)// 2  
        if v[meio] == item:  
            return True  
        elif item < v[meio]:  
            return buscaRecurativa(v,item,inicio,meio- 1)  
        else:  
            return buscaRecurativa(v,item,meio+ 1,fim)  
    else:  
        return False
```

 // Resulta em um valor inteiro

 Olha na esquerda do elemento do meio

 Olha na direita do elemento do meio

```
v=[3,15,70,80,150,461]  
print( buscaRecurativa(v, 700,0, 5))
```

## Pergunta-se

Suponha que você tenha a lista ordenada [3, 5, 6, 8, 11, 12, 14, 15, 17, 18] e que você esteja usando o algoritmo recursivo da busca binária. Qual grupo de números mostra corretamente a sequência de comparações usadas para encontrar a chave 8?

A. 11, 5, 6, 8

B. 12, 6, 11, 8

C. 3, 5, 6, 8

D. 18, 12, 6, 8

# Exercício

Faça um programa que lê diversos números inteiros, ordenados, e depois realiza uma busca binária nesse vetor. O programa deve contar quantas comparações foram realizadas durante a busca.

A busca binária deve ser feita em um subprograma.

## Entradas:

1. Quantidade de números a serem armazenados no vetor.
2. Vários números inteiros, **em ordem crescente**, para armazenar no vetor.
3. Número inteiro a ser buscado no vetor.

## Saídas:

1. O índice do elemento procurado no vetor. Caso o valor não seja encontrado deve ser impresso -1.
2. O número de comparações realizadas entre elementos do vetor e o elemento procurado que foram necessárias para encontrar o valor.

Exemplo de entrada
5 1 2 3 4 5 3
Exemplo de saída
2 1

# Exercício

Dado um vetor com elementos ordenados, elaborar um programa para realizar a busca. Utilizar para isso: busca sequencial e binária.

## Entradas:

1. A quantidade de números que devem ser lidos,
2. Vários números (inteiros, em ordem crescente, pode haver repetição de números),
3. um número a ser procurado.

## Saídas:

1. A posição do número procurado no vetor,
2. O número de comparações necessárias na busca sequencial,
3. O número de comparações necessárias na busca binária.

O programa deve indicar que não encontrou o elemento procurado usando **-1** como posição do elemento.

Exemplo de entrada	Exemplo de saída
10 0 1 2 3 4 5 6 7 8 9 9	9 10 4

# Busca binária em lista - recursiva python

*parâmetros: vetor, valor de busca*

```
def buscaRekursivaPy(v,item):  
    if len(v)==0:  
        return False  
    else:  
        meio=len(v)//2  
        if v[meio] == item:  
            return True  
        elif item < v[meio]:  
            return buscaRekursivaPy(v[:meio-1],item)  
        else:  
            return buscaRekursivaPy(v[meio+1:],item)
```

*posições de 0..meio-1*

*posições de meio+1 até fim*