



# Algoritmos e Estrutura de Dados II

*Aula 04*  
*Vetores Não Ordenados*  
*Busca Sequencial*

Prof. Dr. Dilermando Piva Jr  
2º Semestre - CDN





# Algoritmos de Pesquisa

Ou algoritmos de busca...



# Onde está aquela blusa?





# Listas Lineares

- fácil manipulação
  - agrupa informações referentes a um conjunto de elementos que se relacionam entre si
- Uma lista linear ou tabela é um conjunto de  $n$  elementos  $L[0], L[1], \dots, L[n-1]$  tais que
  - $n > 0$ , e  $L[0]$  é o primeiro elemento
  - para  $0 < k < n$ ,  $L[k]$  é precedido por  $L[k-1]$



# Listas Lineares

- Operações: busca, inclusão e remoção
  - outras operações:
    - alteração de um elemento na lista
    - combinação de duas ou mais listas
    - ordenação
  - Casos particulares:
    - remoção e inserção apenas nas extremidades - deque
    - inserção/remoção em um único extremo - pilha
    - inserções e um extremo e remoções no outro - fila
- Alocação: sequencial ou encadeada



# Listas Sequenciais (tipo list)

- Alocação sequencial de memória
  - endereço do  $(j+1)$ -ésimo elemento se encontra a uma unidade de armazenamento  $j$ -ésimo elemento
- Representação e acesso
  - $i$ -ésimo elemento:  $L[i]$
- Cada elemento pode ser formado por campos
  - uma chave  $k[i]$  está associada ao nó  $L[i]$
  - a lista é dita classificada ou ordenado por chave quando:  
se  $i < j$  então  $k[i]$  precede  $k[j]$



# Busca Sequencial

- busca em uma lista sequencial
  - ordenada pelas suas chaves
  - não ordenada



# Busca Sequencial: Procure o mínimo

PROBLEMA: Encontre o menor valor de uma lista e retorne seu índice

Código Python do algoritmo, na função `minimo()`:

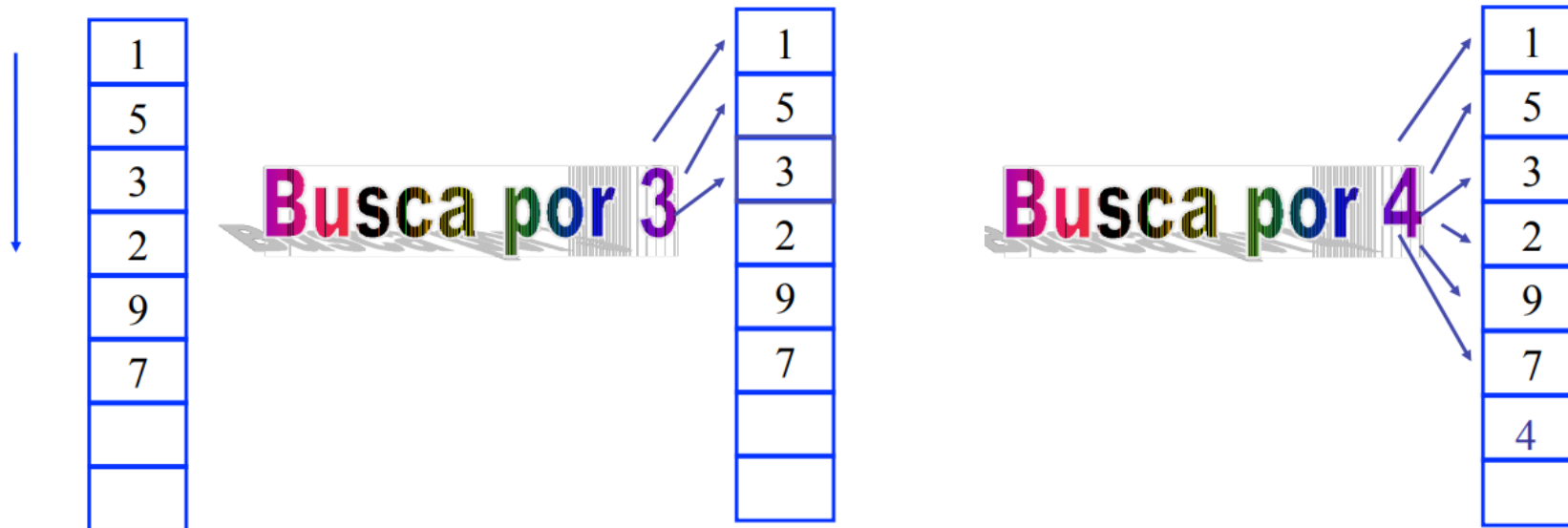
```
def minimo (lista):  
    """Retorna o índice do item mínimo."""  
    indice_min = 0  
    indice_atual = 1  
  
    while indice_atual < len(lista):  
        if lista[indice_atual] < lista[indice_min]:  
            indice_min = indice_atual  
        indice_atual += 1  
    return indice_min
```



# Busca Sequencial: Busca pelo valor v

PROBLEMA: Encontre o valor v de uma lista e retorne seu índice.  
Caso não encontre, retorne -1

Nada foi especificado: consideremos a lista com elementos não ordenados





# Pesquisa sequencial de uma lista

Código Python para uma função de pesquisa sequencial:

```
def busca_sequencial(v, lista):  
    """Retorna a posição do item-alvo se encontrado, ou -1 caso contrário."""  
    posicao = 0  
    while posicao < len(lista):  
        if v == lista[posicao]:  
            return posicao  
        posicao += 1  
    return -1
```



# Pesquisa sequencial de uma lista

Código Python para uma função de pesquisa sequencial:

```
def busca_sequencial(v, lista):  
    """Retorna a posição do item-alvo se encontrado, ou -1 caso contrário."""  
    posicao = 0  
    while posicao < len(lista):  
        if v == lista[posicao]:  
            return posicao  
        posicao += 1  
    return -1
```

Qual a Complexidade?

$O(n)$



# Pesquisa sequencial de uma lista

Código Python para uma função de pesquisa sequencial:

```
def busca_sequencial(v, lista):  
    """Retorna a posição do item-alvo se encontrado, ou -1 caso contrário."""  
    posicao = 0  
    while posicao < len(lista):  
        if v == lista[posicao]:  
            return posicao  
        posicao += 1  
    return -1
```

Qual a Complexidade?

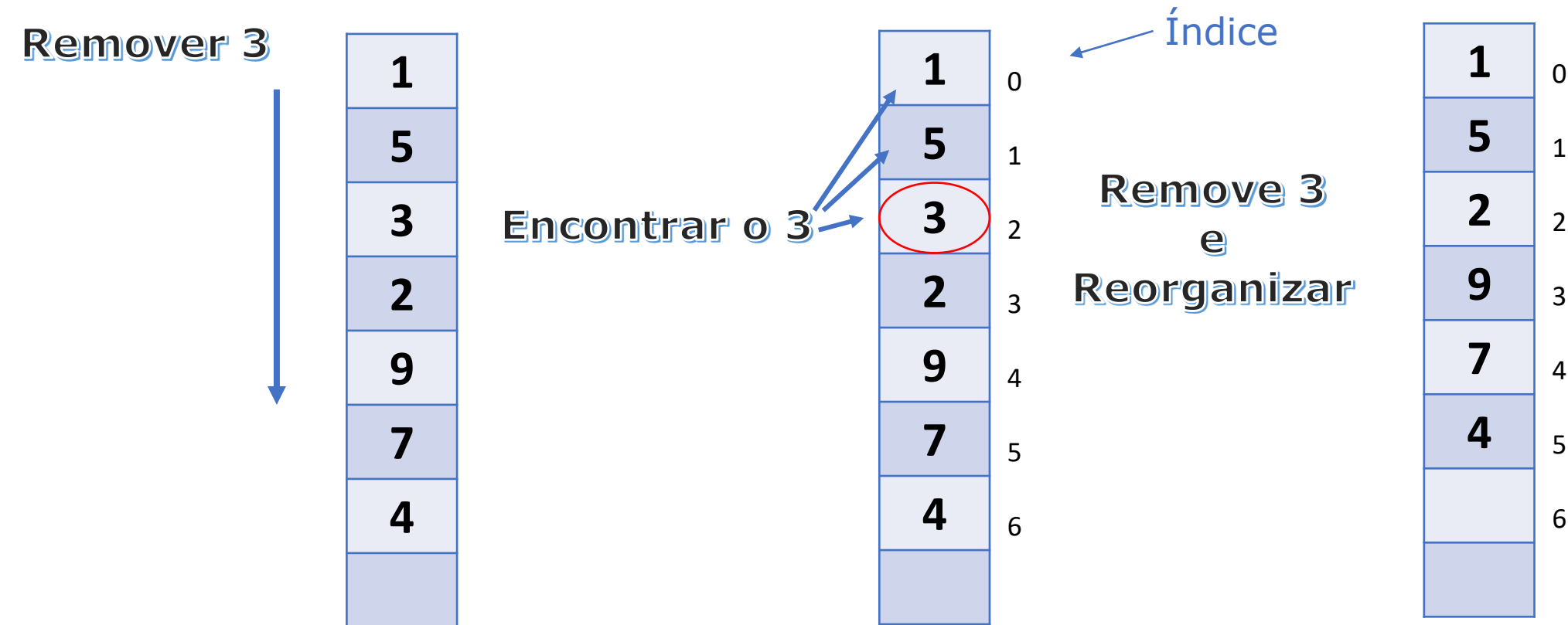
**$O(n)$**



# Remoção em Lista Sequencial: valor v

PROBLEMA: Remover o valor v de uma lista sequencial e reorganize toda a lista. Caso não encontre, retorne -1. Caso seja removido com sucesso, retorna v

Nada foi especificado: consideremos a lista com elementos não ordenados





# Remoção em Lista Sequencial: valor v

PROBLEMA: Remover o valor v de uma lista sequencial e reorganize toda a lista. Caso não encontre, retorne -1. Caso seja removido com sucesso, retorna v

```
def remove_lista_desordenada(v, lista):  
    """Remove o valor 'v' da lista desordenada 'lista'."""  
    if len(lista) == 0:  
        # A lista está vazia  
        return -1  
  
    indice = busca_sequencial(v, lista)  
  
    if indice != -1:  
        elemento = lista[indice]  
        for i in range(indice, len(lista) - 1):  
            lista[i] = lista[i + 1]  
        lista.pop() # remove o último elemento que está duplicado  
        return elemento  
    else:  
        return "Elemento não encontrado"
```



# Remoção em Lista Sequencial: valor v

PROBLEMA: Remover o valor v de uma lista sequencial e reorganize toda a lista. Caso não encontre, retorne -1. Caso seja removido com sucesso, retorna v

```
def remove_lista_desordenada(v, lista):  
    """Remove o valor 'v' da lista desordenada 'lista'."""  
    if len(lista) == 0:  
        # A lista está vazia  
        return -1  
  
    indice = busca_sequencial(v, lista)  
  
    if indice != -1:  
        elemento = lista[indice]  
        for i in range(indice, len(lista) - 1):  
            lista[i] = lista[i + 1]  
        lista.pop() # remove o último elemento que está duplicado  
        return elemento  
    else:  
        return "Elemento não encontrado"
```

Complexidade?

$O(n)$



# Remoção em Lista Sequencial: valor v

PROBLEMA: Remover o valor v de uma lista sequencial e reorganize toda a lista. Caso não encontre, retorne -1. Caso seja removido com sucesso, retorna v

```
def remove_lista_desordenada(v, lista):  
    """Remove o valor 'v' da lista desordenada 'lista'."""  
    if len(lista) == 0:  
        # A lista está vazia  
        return -1  
  
    indice = busca_sequencial(v, lista) O(n)  
  
    if indice != -1:  
        elemento = lista[indice]  
        for i in range(indice, len(lista) - 1): O(n)  
            lista[i] = lista[i + 1]  
        lista.pop() # remove o último elemento que está duplicado  
        return elemento  
    else:  
        return "Elemento não encontrado"
```

Complexidade?  
 $O(n) * (n)$   
 **$O(n^2)$**



# VAMOS PARA A PRÁTICA ?!!!

