Aula 03

Vetores não-ordenados - Busca Sequencial

Algoritmos e Estrutura de Dados II

2° Semestre - CDN

Prof. Dr. Dilermando Piva Jr.

Conteúdo Programático - Planejamento

Semana	Data	Temas/Atividades
1	07/08	Acolhimento e Boas-vindas! Introdução a Disciplina. Formas de Avaliação e Percurso Pedagógico.
2	14/08	Tipo de dado abstrato. Introdução a Estrutura de Dados.
3	21/08	Complexidade de Algoritmos
4	28/08	Vetores não-Ordenados e busca sequencial
5	04/09	Vetores Ordenados e busca binária
6	11/09	Revisão de Programação Orientada a Objetos (POO)
7	18/09	Pilhas
8	25/09	Filas
9	02/10	Listas encadeadas
10	09/10	Recursão
11	*16/10*	***Primeira Avaliação Formal. (P1).*** *Correção da Avaliação após o intervalo.*
12	18/10	Algoritmos de Ordenação
13	23/10	Algoritmos de Ordenação
14	30/10	Árvores
15	06/11	Grafos
16	13/11	***Segunda Avaliação Formal (P2).*** Correção da Avaliação após o intervalo
17	27/11	***Apresentação PI do curso de CDN***
18	*04/12*	Tabela Hash (tabela de espalhamento) – Tópico extra.
19	*11/12*	***Exame / Avaliação Substitutiva.*** Correção da Avaliação após o intervalo. Finalização Disciplina
20	18/12	Finalização da disciplina.

Cenário:
Lista de Compras

Anota o itens à medida que lembra

Imagem Gerada por IA – ChatGPT 4o

Onde está aquela blusa?

Algoritmos de Pesquisa

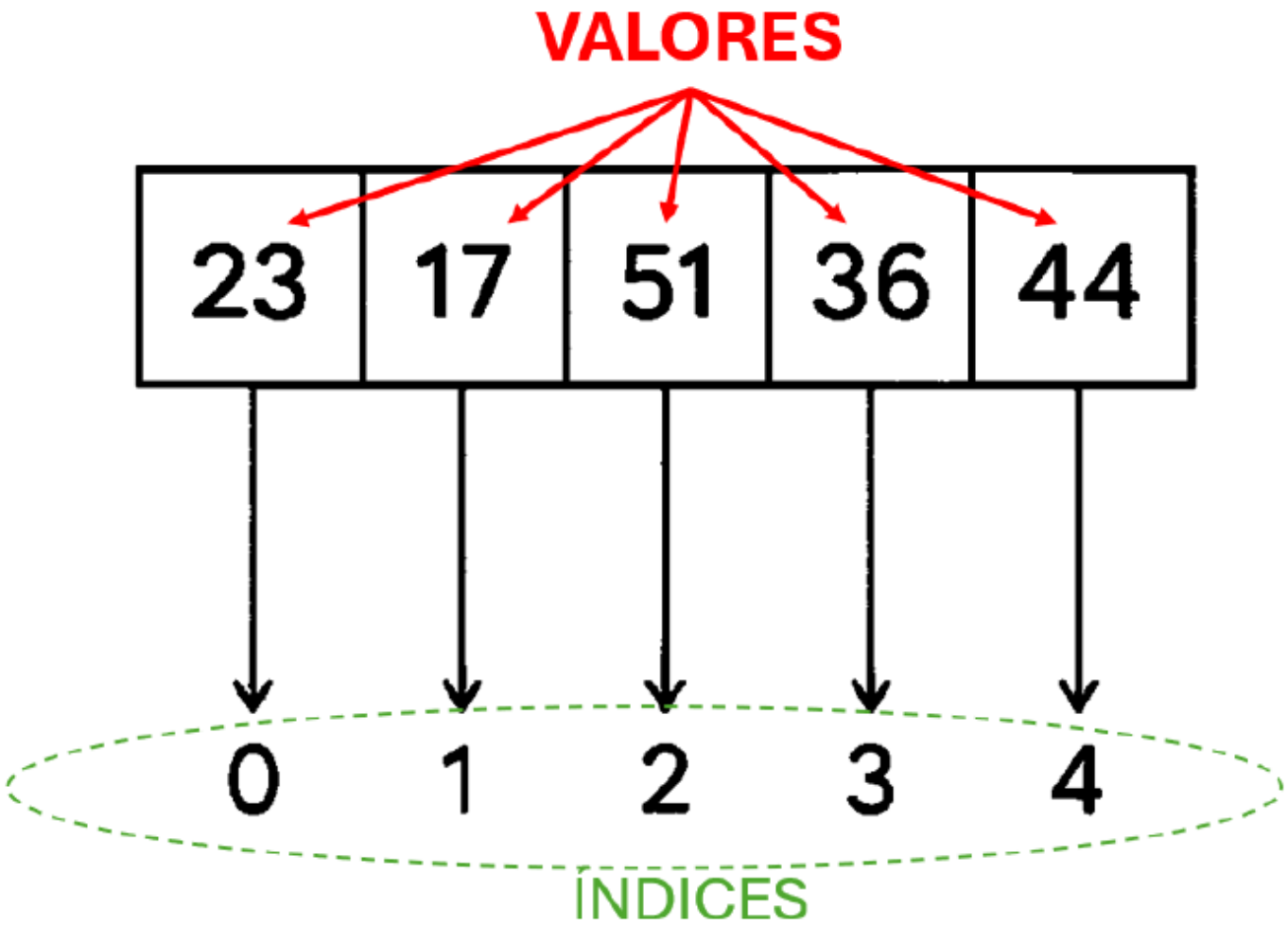
Ou algoritmos de busca...

Listas Lineares / Vetores

- 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], ..., 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 / Vetores



Listas Lineares / Vetores

- 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 Lineares / Vetores

Exemplo cotidiano:

Pense em uma prateleira com vários livros em sequência.

Cada livro tem uma posição definida na estante (podemos numerar os espaços da esquerda para a direita).

Essa disposição linear permite que você acesse diretamente o livro na posição *n* contando a partir da esquerda.

Um **vetor** é análogo a essa prateleira: podemos pegar o livro número 3 diretamente.

Agora, se os livros não estiverem organizados alfabeticamente ou por assunto, encontrar um título específico exigirá examinar um por um, na ordem da prateleira – esse é o princípio da busca sequencial.

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:

$$\text{se } i < j \text{ então } k[i] \text{ precede } k[j]$$

Busca Sequencial

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

Busca Sequencial

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

Atividade com IA

- Vamos saber mais sobre:
  - Quais os passos para encontrar um elemento em uma lista (vetor) não ordenado?

- Faça individualmente, e depois compartilhe com o seu colega esses conceitos.

  O que caracteriza um vetor não ordenado?

  Quais são os passos necessários para encontrar um valor dentro desse vetor?

  O que acontece se o elemento estiver presente?

  O que acontece se o elemento não estiver presente?

  Qual é a eficiência (tempo de execução) desse processo de busca?

Possíveis Respostas:

1. O que caracteriza um vetor não ordenado?

- Um vetor não ordenado é uma estrutura de dados na qual os elementos não seguem nenhuma ordem específica (nem crescente, nem decrescente, nem outra).
- Isso significa que não é possível usar estratégias de busca mais eficientes, como a busca binária.
- A única maneira de localizar um elemento é percorrer os itens um por um até encontrá-lo ou chegar ao fim do vetor.

Possíveis Respostas:

2. Quais são os passos necessários para encontrar um valor dentro desse vetor?

1. Definir o valor que se deseja encontrar (chamado chave de busca).
2. Começar a percorrer o vetor do primeiro elemento até o último.
3. Em cada posição, comparar o valor armazenado com a chave de busca.
4. Se o valor for encontrado, parar a busca e retornar a posição.
5. Se chegar ao final do vetor sem encontrar, concluir que o elemento não está presente.

Possíveis Respostas:

3. O que acontece se o elemento estiver presente?

O algoritmo retorna a posição (índice) em que o elemento foi encontrado.

Em alguns casos, pode retornar também uma mensagem de sucesso ou o próprio valor.

4. O que acontece se o elemento não estiver presente?

O algoritmo percorre todo o vetor e, ao não encontrar o valor, retorna uma indicação de “não encontrado” (por exemplo, -1 ou `None`).

Possíveis Respostas:

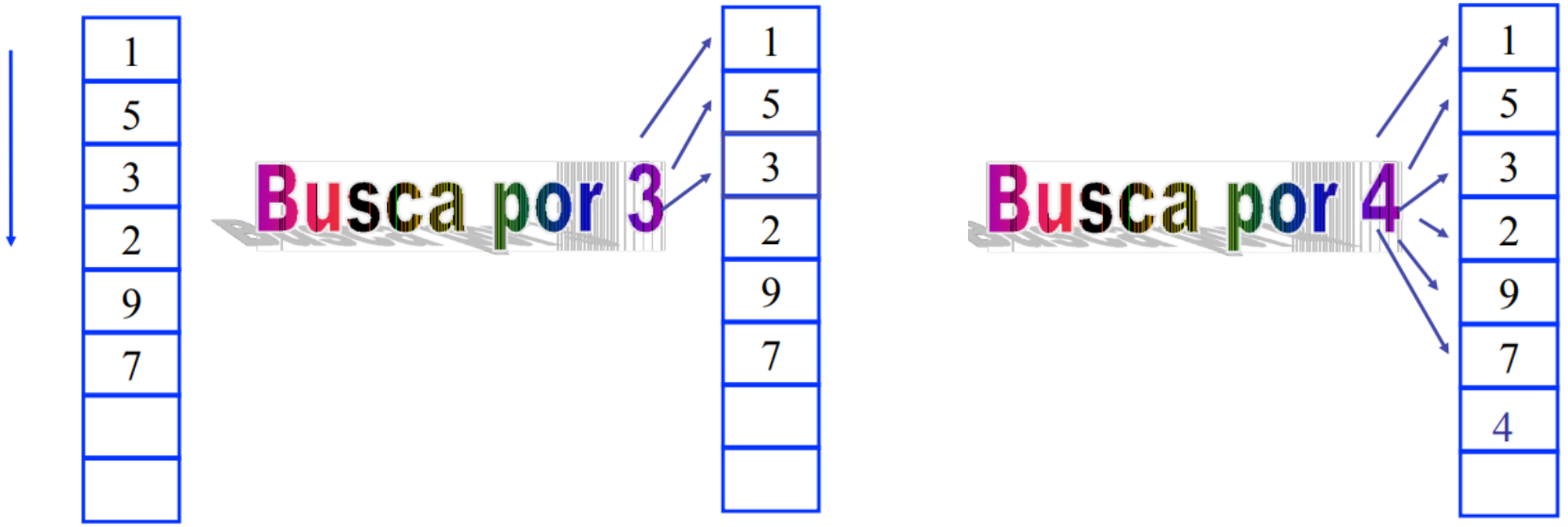
5. Qual é a eficiência (tempo de execução) desse processo de busca?

- **Melhor caso**: o elemento está logo na primeira posição → 1 comparação → $O(1)$.
- **Pior caso**: o elemento está na última posição ou não existe no vetor → n comparações → $O(n)$.
- **Caso médio**: aproximadamente n/2 comparações, **mas a complexidade assintótica continua sendo $O(n)$.**

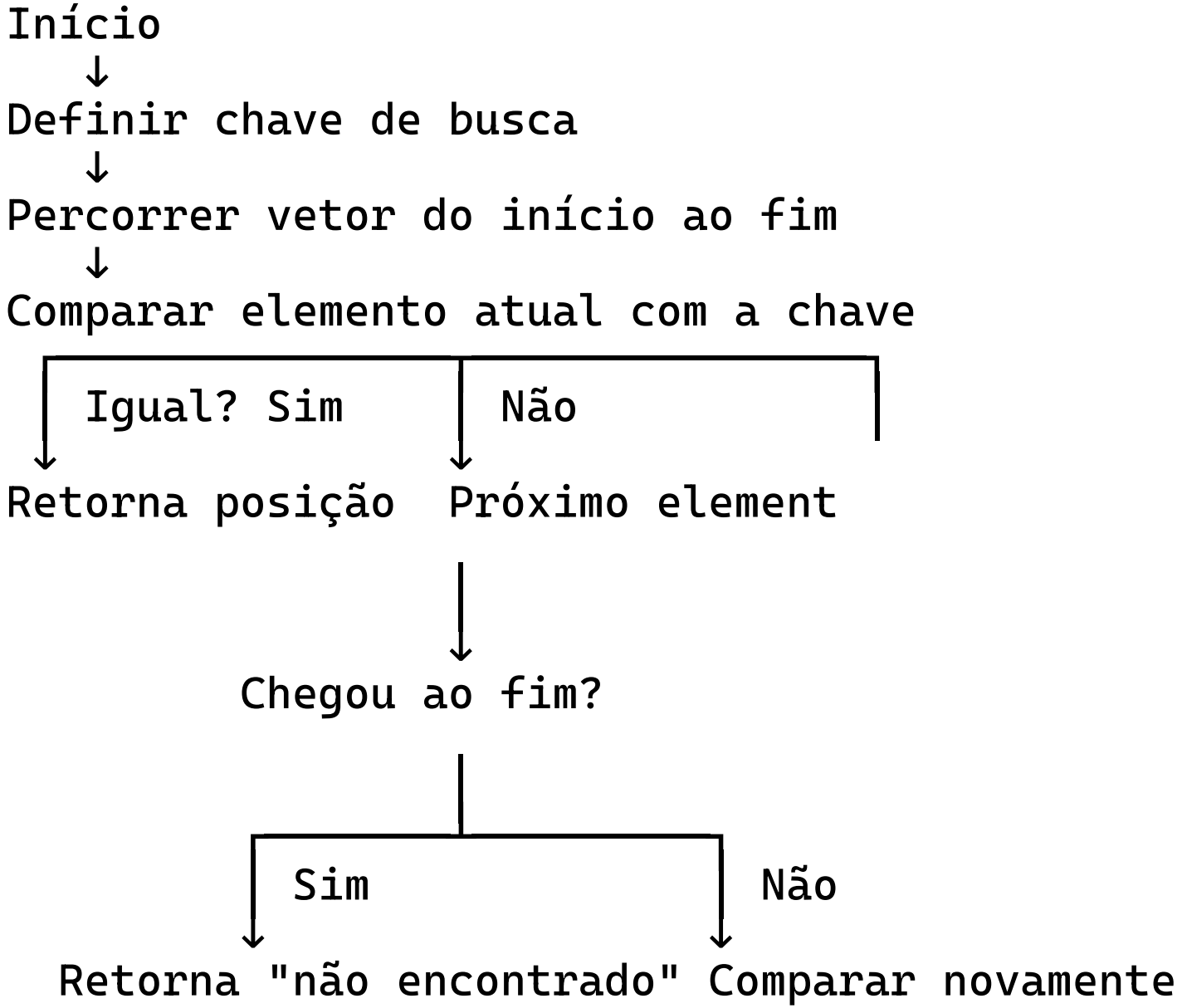
Busca Sequencial: Busca pelo valor v

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



Fluxo...


Início
Definir chave de busca
Percorrer vetor do início ao fim
Comparar elemento atual com a chave
Igual? Sim
Não
Retorna posição
Próximo element
Chegou ao fim?
Sim
Não
Retorna "não encontrado"
Comparar novamente

Pesquisa sequencial de uma lista (Solução 1)

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 (Solução 2)

Código Python para uma função de pesquisa sequencial:

```
def busca_sequencial(vetor, valor):
    for i in range(len(vetor)):
        if vetor[i] == valor:
            return i                    # retorna o índice onde encontrou
    return -1                           # retorna -1 se não encontrou


# Exemplo de uso
vetor = [8, 3, 10, 1, 7]
print(busca_sequencial(vetor, 10))  # saída: 2
print(busca_sequencial(vetor, 5))   # saída: -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 ( )

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)

Busca Sequencial: Procure o mínimo

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

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

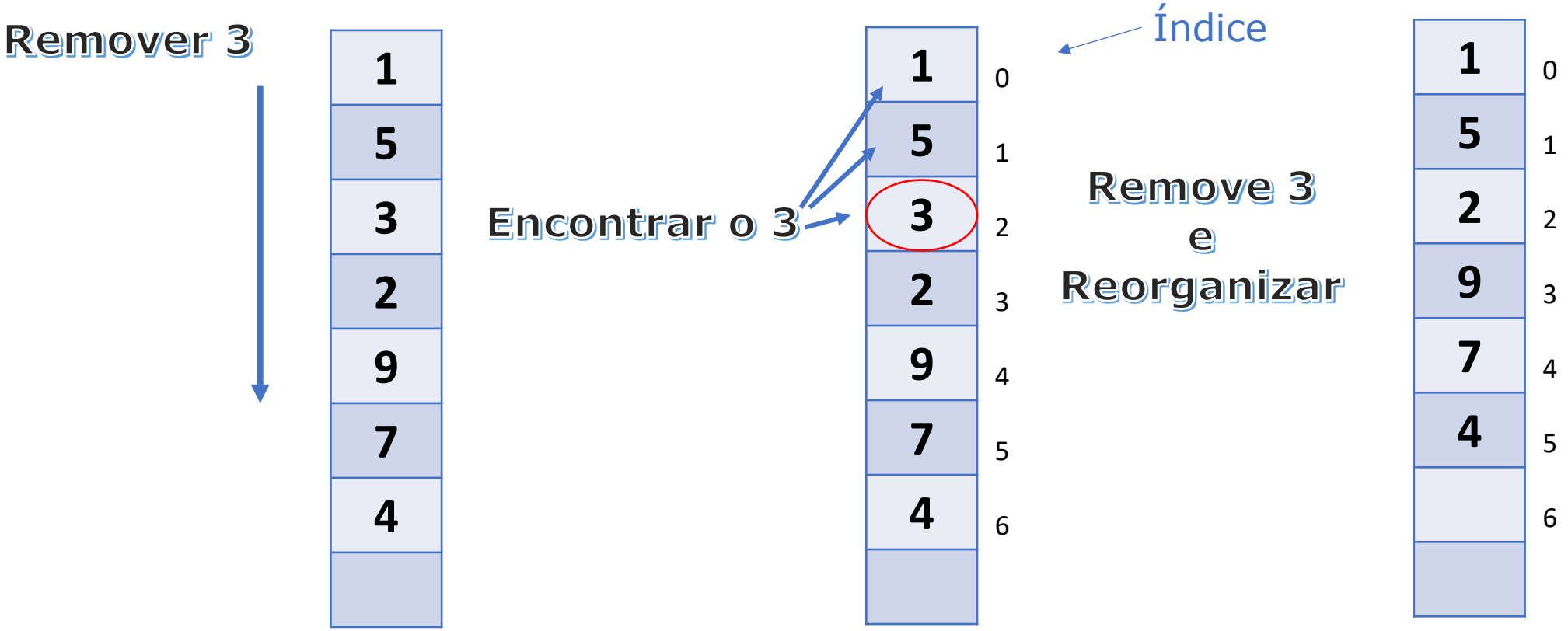
```
def minimo (lista):
    """Retorna o índice do item mínimo."""
    indice_min = 0
    indice_atual = 1

    while indice_atual < len(lista):
        if lista[indice_atual] < lista[indice_min]:
            indice_min = indice_atual
        indice_atual += 1
    return indice_min
```

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 ( )

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"
```

O(n)

O(n)

Complexidade?
O(n) + (n)
O (n)

VAMOS PARA A PRÁTICA ?!!!

Exercícios...

1. (ENADE): Considere um algoritmo de busca sequencial que procura por um determinado valor em um vetor de tamanho n. Sobre a complexidade desse algoritmo, assinale a alternativa **CORRETA**:

A. Melhor caso em tempo O(n) e pior caso em tempo $O(n^2)$.
B. Melhor caso em tempo O(1) e pior caso em tempo O(n).
C. Melhor caso e pior caso ambos O(log n).
D. A complexidade independe de n (tempo constante, O(1)).

Exercícios...

1. (ENADE): Considere um algoritmo de busca sequencial que procura por um determinado valor em um vetor de tamanho n. Sobre a complexidade desse algoritmo, assinale a alternativa **CORRETA**:

A. Melhor caso em tempo O(n) e pior caso em tempo $O(n^2)$.
B. Melhor caso em tempo O(1) e pior caso em tempo O(n).
C. Melhor caso e pior caso ambos O(log n).
D. A complexidade independe de n (tempo constante, O(1)).

Exercícios...

2. (TEÓRICA): Suponha que você tenha uma lista não ordenada com 100 nomes e deseja verificar se o nome "Mariana" está presente. Sobre o algoritmo de busca sequencial nesse contexto, é **correto afirmar** que:

A. No pior caso, será necessário comparar "Mariana" com **todos** os 100 nomes da lista.
B. Caso "Mariana" não esteja na lista, o algoritmo irá interromper a busca na metade, por eficiência.
C. Se "Mariana" estiver na primeira posição, ainda assim serão verificadas todas as 100 posições para confirmar duplicatas.
D. A busca sequencial não funciona com listas de strings, apenas com números, devido às comparações.

Exercícios...

2. (TEÓRICA): Suponha que você tenha uma lista não ordenada com 100 nomes e deseja verificar se o nome "Mariana" está presente. Sobre o algoritmo de busca sequencial nesse contexto, é **correto afirmar** que:

A. No pior caso, será necessário comparar "Mariana" com **todos** os 100 nomes da lista.
B. Caso "Mariana" não esteja na lista, o algoritmo irá interromper a busca na metade, por eficiência.
C. Se "Mariana" estiver na primeira posição, ainda assim serão verificadas todas as 100 posições para confirmar duplicatas.
D. A busca sequencial não funciona com listas de strings, apenas com números, devido às comparações.

Exercícios...

3. (TEÓRICA): Considere a implementação clássica da busca sequencial que retorna o índice da primeira ocorrência do valor alvo ou -1 se não encontrar. Dada uma lista `L = [7, 12, 5, 12, 8]`, se chamarmos a função `busca_sequencial(12, L)`, qual será o resultado retornado?

A. 1
B. 2
C. 3
D. -1

Exercícios...

3. (TEÓRICA): Considere a implementação clássica da busca sequencial que retorna o índice da primeira ocorrência do valor alvo ou -1 se não encontrar. Dada uma lista `L = [7, 12, 5, 12, 8]`, se chamarmos a função `busca_sequencial(12, L)`, qual será o resultado retornado?

A. 1

B. 2

C. 3

D. -1

Exercícios...

4. (PRÁTICA): Implemente uma função em Python chamada `buscar_cliente(nome, lista_clientes)` que realiza busca sequencial em uma lista de nomes de clientes. Essa função deve retornar o índice onde o `nome` foi encontrado ou `-1` caso o nome não esteja na `lista_clientes`. Em seguida, escreva um pequeno código de exemplo que utilize essa função para buscar por pelo menos um nome que exista e um que não exista na lista, exibindo mensagens adequadas.

Exercícios...

```
def buscar_cliente(nome, lista_clientes):
    for i in range(len(lista_clientes)):
        if lista_clientes[i] == nome:
            return i  # encontrou, retorna a posição
    return -1  # não encontrou, retorna -1

# Lista de exemplo e testes da função
clientes = ["Ana", "Bruno", "Carlos", "Daniel", "Elisa"]
busca1 = "Carlos"
busca2 = "Fernanda"

pos1 = buscar_cliente(busca1, clientes)
if pos1 != -1:
    print(f"Cliente '{busca1}' encontrado na posição {pos1}.")
else:
    print(f"Cliente '{busca1}' não encontrado na lista.")

pos2 = buscar_cliente(busca2, clientes)
if pos2 != -1:
    print(f"Cliente '{busca2}' encontrado na posição {pos2}.")
else:
    print(f"Cliente '{busca2}' não encontrado na lista.")
```