Educação Profissional Paulista

Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Lógica de programação e algoritmos

Prática de busca e ordenação

Aula 2 - Exercícios práticos de busca e ordenação

Código da aula: [DADOS]ANO1C3B4S25A2

Mapa da Unidade 1 Componente 4

Lógica de programação e algoritmos

semana 25

Você está aqui!
Prática de busca e ordenação

semana 27

Algoritmos de contagem e acumulação

semana 28

Pilhas e filas

Lógica de programação e algoritmos

Mapa da Unidade 1 Componente 4

Você está aqui!

25

Prática de busca e ordenação

Aula 2

Código da aula:
[DADOS]ANO1C3B4S25A2

Objetivos da aula

- Introduzir e praticar conceitos sobre algoritmos de busca e ordenação.

Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou internet.

Duração da aula

50 minutos.

Competências técnicas

- Identificar e resolver problemas relacionados a dados e análises;
- Compreender e dominar técnicas de manipulação de dados;

Competências socioemocionais

- Adaptar-se a novas tecnologias, técnicas e tendências sem perder o foco, as metas e os objetivos da organização.
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais colaborando com colegas, gestores e clientes.

Construindo o **conceito**

Exemplos práticos

Exemplo 1: Busca linear em lista de números

```python
# Lista de números
numeros = [34, 67, 23, 90, 12, 45, 78]

# Função de busca linear
def busca_linear(lista, elemento):
    for i in range(len(lista)):
        if lista[i] == elemento:
            return i
    return -1

# Teste
elemento = 90
indice = busca_linear(numeros, elemento)
print(f"Elemento {elemento} encontrado na posição {indice}" if indice != -1 else "Elemento não encontrado")
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

Exemplos práticos

Exemplo 2: Busca linear em lista de palavras

```
# Lista de palavras
palavras = ["casa", "carro", "livro", "telefone", "computador"]

# Função de busca linear (usando a mesma função do exercício anterior)

# Teste
elemento = "livro"
indice = busca_linear(palavras, elemento)
print(f"Palavra '{elemento}' encontrada na posição {indice}" if indice != -1 else "Palavra não encontrada")
```

Construindo
o **conceito**

Exemplos práticos

Exemplo 3: Busca linear em registro de alunos

```
# Lista de alunos
alunos = ["Ana", "Bruno", "Carlos", "Daniela", "Fernanda"]

# Função de busca linear (usando a mesma função do exercício anterior)

# Teste
elemento = "Carlos"
indice = busca_linear(alunos, elemento)
print(f"Aluno '{elemento}' encontrado na posição {indice}" if indice != -1 else "Aluno não encontrado")
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

Exemplos práticos

Exemplo 4: Busca binária em lista de números

```python
# Lista de números ordenada
numeros = [12, 23, 34, 45, 67, 78, 90]

# Função de busca binária
def busca_binaria(lista, elemento):
    inicio = 0
    fim = len(lista) - 1
    while inicio <= fim:
        meio = (inicio + fim) // 2
        if lista[meio] == elemento:
            return meio
        elif lista[meio] < elemento:
            inicio = meio + 1
        else:
            fim = meio - 1
    return -1

# Teste
elemento = 67
indice = busca_binaria(numeros, elemento)
print(f"Elemento {elemento} encontrado na posição {indice}" if indice != -1 else "Elemento não encontrado")
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

Exemplos práticos

Exemplo 5: Busca binária em lista de palavras

```
# Lista de palavras ordenada
palavras = ["banana", "computador", "casa", "livro", "telefone"]

# Função de busca binária (usando a mesma função do exercício anterior)

# Teste
elemento = "livro"
indice = busca_binaria(palavras, elemento)
print(f"Palavra '{elemento}' encontrada na posição {indice}" if indice != -1 else "Palavra não encontrada")
```

Construindo o **conceito**

Exemplos práticos

Exemplo 6: Ordenação por seleção em lista de números

```
# Lista de números desordenada
numeros = [34, 67, 23, 90, 12, 45, 78]

# Função de ordenação por seleção
def ordenacao_por_selecao(lista):
    n = len(lista)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if lista[j] < lista[min_idx]:
                min_idx = j
        lista[i], lista[min_idx] = lista[min_idx], lista[i]

# Teste
ordenacao_por_selecao(numeros)
print("Lista ordenada por seleção:", numeros)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

Exemplos práticos

Exemplo 7: Ordenação por seleção em lista de palavras

```
# Lista de palavras desordenada
palavras = ["casa", "carro", "livro", "telefone", "computador"]

# Função de ordenação por seleção (usando a mesma função do exercício anterior)

# Teste
ordenacao_por_selecao(palavras)
print("Lista ordenada por seleção:", palavras)
```

Construindo o **conceito**

Exemplos práticos

Exemplo 8: Ordenação por inserção em lista de números

```
# Lista de números desordenada
numeros = [34, 67, 23, 90, 12, 45, 78]

# Função de ordenação por inserção
def ordenacao_por_insercao(lista):
    for i in range(1, len(lista)):
        chave = lista[i]
        j = i - 1
        while j >= 0 and lista[j] > chave:
            lista[j + 1] = lista[j]
            j -= 1
        lista[j + 1] = chave

# Teste
ordenacao_por_insercao(numeros)
print("Lista ordenada por inserção:", numeros)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo
o **conceito**

Exemplos práticos

Exemplo 9: Ordenação por inserção em lista de palavras

```
# Lista de palavras desordenada
palavras = ["casa", "carro", "livro", "telefone", "computador"]

# Função de ordenação por inserção (usando a mesma função do exercício anterior)

# Teste
ordenacao_por_insercao(palavras)
print("Lista ordenada por inserção:", palavras)
```

Colocando em **prática**

Exercício: Comparação de algoritmos de busca e ordenação

Objetivo:
Proporcionar a prática direta e simplificada de algoritmos de busca e ordenação, reforçando a aplicação desses conceitos em listas pequenas.

Descrição do exercício:
Formem duplas para aplicar a busca linear e a ordenação por inserção em uma lista de números pequena e simples. É preciso completar e testar as funções de ordenação e busca para entender melhor como cada algoritmo funciona.

Lista de números: [30, 10, 20, 40, 5].

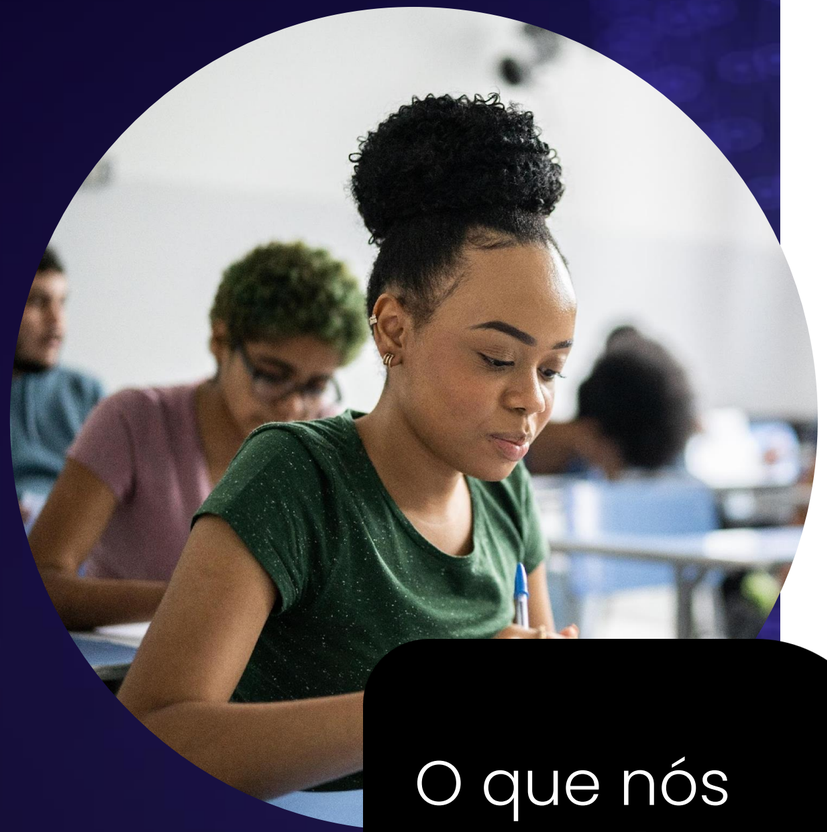
Tarefas:
- Ordenar a lista de números usando a ordenação por inserção.
- Encontrar o número 20 na lista de números usando busca linear.

20 minutos

Duplas

Código de programação

Diálogo

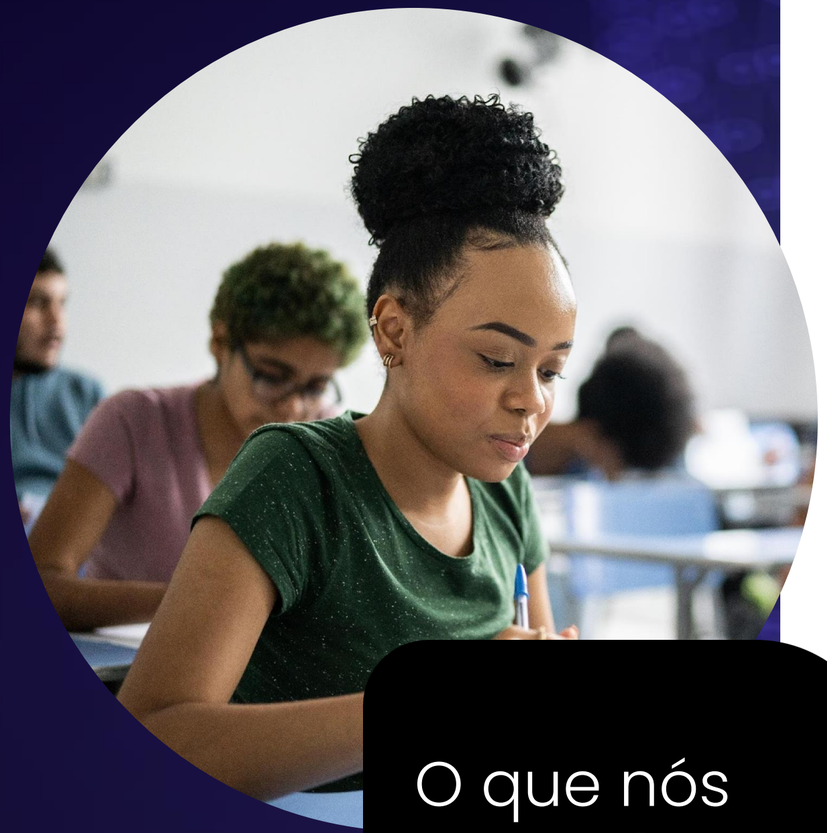


© Getty Images

O que nós **aprendemos hoje?**

Nuvem de palavras

Diálogo



© Getty Images

O que nós **aprendemos hoje?**

Então ficamos assim...

1. Focamos em exercícios práticos de busca e ordenação. Aplicamos a busca linear e ordenação por inserção, em lista simples de números, reforçando a teoria por meio da prática;

2. Os exercícios incluíram a implementação e teste da ordenação por inserção, que organiza uma lista de números em ordem crescente e da busca linear, utilizada para encontrar um número específico na lista;

3. A sessão prática permitiu observar diretamente o funcionamento dos algoritmos, destacando a eficácia da ordenação por inserção para pequenas listas e a simplicidade da busca linear em encontrar elementos.

Saiba mais

Está pronto para dominar a área de data science e se tornar um profissional de alta demanda no mercado?

A formação de Python para data science da Alura é a sua chance de mergulhar nesse universo, com um guia completo e um passo a passo que te prepara com as habilidades essenciais para enfrentar os desafios da área.

ALURA. Formação Python para Data Science, [s.d.]. Disponível em: https://www.alura.com.br/formacao-data-science-python/. Acesso em: 11 jul. 2024.

Referências da aula

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. *Lógica de programação*: a construção de algoritmos e estruturas de dados com aplicações em Python. São Paulo: Pearson; Porto Alegre: Bookman, 2022.

Educação Profissional Paulista

Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO