

Tecnólogo Ciência de Dados

Guilherme Giacomini Teixeira

título do trabalho:

Relatório de Aula prática

Balneário Camboriú - SC

2025

Balneário Camboriú - SC

2025

Guilherme Giacomini Teixeira

Aula 4 - Árvores Avl:

Relatório de Aula prática

Aula prática de linguagem de programação apresentado como requisito parcial para a obtenção de média semestral no curso Analise e Desenvolvimento de Sistemas.

Orientador: Luana Gomes de Souza  
Tutor: Luana Gomes de Souza

SUMÁRIO

[1 INTRODUÇÃO 3](#_Toc129809424)

[2 DESENVOLVIMENTO 4](#_Toc129809425)

[3 RESULTADOS](#_Toc129809426) 6

[4 CONCLUSÃO](#_Toc129809427) 7

[5 REFERÊNCIAS](#_Toc129809428) 8

# INTRODUÇÃO

Este relatório apresenta a atividade solicitada na disciplina XXXXXX, realizada no XXXX semestre do curso de Análise e Desenvolvimento de Sistemas.

O objetivo da atividade foi criar um programa que calcule IMC (índice de massa corpórea) utilizando a ferramenta Google Clound Shell, e utilizando a linguagem de programação Python.

Neste relatório, serão descritos os passos seguidos para o desenvolvimento do projeto.

# DESENVOLVIMENTO

1. Primeiramente vai explicar sobre a linguagem Python

"""

Aqui está a atividade/trabalho da Unidade 2 da disciplina de Estrutura de Dados.

Deixarei comentários no código para deixar mais claro tanto o raciocínio da

implementação quanto a lógica por trás das Árvores AVL.

"""

*# ---------------------------------*

*# ATIVIDADE PROPOSTA*

*# ---------------------------------*

*# Você foi contratado como um desenvolvedor de software para um novo jogo de*

*# Pokémon e recebeu a tarefa de implementar um sistema eficiente de gerenciamento*

*# de Pokémon utilizando uma Árvore AVL. Cada Pokémon no jogo é caracterizado*

*# por um nome e um valor de força, que é um número inteiro.*

*# ---------------------------------*

*# Objetivos do Exercício*

*# ---------------------------------*

*# 1. Implementação da Árvore AVL:*

*#    - Implemente uma Árvore AVL que armazene informações sobre cada Pokémon.*

*#      Os nós da árvore devem conter o nome do Pokémon e seu valor de força.*

*#      A árvore será ordenada pelo nome.*

*# 2. Funcionalidade de Busca:*

*#    - Desenvolva uma função para buscar Pokémon na árvore pelo nome. Esta função*

*#      deve retornar as informações do Pokémon se ele for encontrado.*

*# 3. Funcionalidade de Listagem:*

*#    - Crie uma função para listar todos os Pokémon da árvore em ordem*

*#      decrescente de força. Isso permitirá que os jogadores vejam os*

*#      Pokémon mais fortes primeiro.*

*# 4. Funcionalidade de Remoção:*

*#    - Implemente uma função para remover Pokémon da árvore por nome, garantindo*

*#      que a árvore se mantenha balanceada após a remoção.*

*# ---------------------------------*

*# CÓDIGO PARA A ATIVIDADE*

*# ---------------------------------*

*# - PokemonNode: A classe que define a estrutura de cada nó da árvore.*

*# - AVLTree: A classe principal que contém toda a lógica da árvore, incluindo:*

*#     - Funções auxiliares para altura e balanceamento.*

*#     - As rotações (\_rotacao\_direita e \_rotacao\_esquerda) que são o coração*

*#       do balanceamento.*

*#     - As funções principais (inserir, remover, buscar,*

*#       listar\_em\_ordem\_decrescente\_forca).*

*# - Programa Principal (if \_\_name\_\_ == "\_\_main\_\_":):*

*#     Uma seção para testar todas as funcionalidades implementadas, mostrando*

*#     que a árvore funciona como esperado.*

class PokemonNode:

    """

    Classe que representa um nó na Árvore AVL.

    Cada nó armazena os dados de um Pokémon e a altura da sub-árvore.

    A chave para a ordenação da árvore será o nome do Pokémon.

    """

    def **\_\_init\_\_**(*self*, *nome*, *forca*):

*self*.nome = *nome*

*self*.forca = *forca*

*self*.altura = 1

*self*.esquerda = None

*self*.direita = None

class AVLTree:

    """

    Classe que implementa a Árvore AVL com as funcionalidades

    de inserção, remoção, busca e listagem de Pokémon.

    """

*# --- Funções Auxiliares da Árvore ---*

    def **get\_altura**(*self*, *no*):

        """Retorna a altura de um nó (0 se o nó for nulo)."""

*if* not *no*:

*return* 0

*return* *no*.altura

    def **get\_balanco**(*self*, *no*):

        """Calcula o fator de balanceamento de um nó."""

*if* not *no*:

*return* 0

*return* *self*.get\_altura(*no*.esquerda) - *self*.get\_altura(*no*.direita)

    def **get\_min\_valor\_no**(*self*, *no*):

        """Encontra o nó com o menor valor (nome) em uma sub-árvore."""

*if* *no* is None or *no*.esquerda is None:

*return* *no*

*return* *self*.get\_min\_valor\_no(*no*.esquerda)

*# --- Rotações da Árvore AVL ---*

    def **\_rotacao\_direita**(*self*, *z*):

        """Executa uma rotação simples à direita."""

        print(f"Executando rotação à direita no nó: {*z*.nome}")

        y = *z*.esquerda

        T3 = y.direita

*# Realiza a rotação*

        y.direita = *z*

*z*.esquerda = T3

*# Atualiza as alturas*

*z*.altura = 1 + max(*self*.get\_altura(*z*.esquerda), *self*.get\_altura(*z*.direita))

        y.altura = 1 + max(*self*.get\_altura(y.esquerda), *self*.get\_altura(y.direita))

*return* y

    def **\_rotacao\_esquerda**(*self*, *z*):

        """Executa uma rotação simples à esquerda."""

        print(f"Executando rotação à esquerda no nó: {*z*.nome}")

        y = *z*.direita

        T2 = y.esquerda

*# Realiza a rotação*

        y.esquerda = *z*

*z*.direita = T2

*# Atualiza as alturas*

*z*.altura = 1 + max(*self*.get\_altura(*z*.esquerda), *self*.get\_altura(*z*.direita))

        y.altura = 1 + max(*self*.get\_altura(y.esquerda), *self*.get\_altura(y.direita))

*return* y

*# --- Funcionalidades Principais ---*

    def **inserir**(*self*, *raiz*, *nome*, *forca*):

        """Insere um novo Pokémon na árvore e a balanceia."""

*# 1. Inserção padrão de uma Árvore de Busca Binária*

*if* not *raiz*:

*return* PokemonNode(*nome*, *forca*)

*elif* *nome* < *raiz*.nome:

*raiz*.esquerda = *self*.inserir(*raiz*.esquerda, *nome*, *forca*)

*else*:

*raiz*.direita = *self*.inserir(*raiz*.direita, *nome*, *forca*)

*# 2. Atualiza a altura do nó ancestral*

*raiz*.altura = 1 + max(*self*.get\_altura(*raiz*.esquerda), *self*.get\_altura(*raiz*.direita))

*# 3. Calcula o fator de balanceamento e aplica as rotações se necessário*

        balanco = *self*.get\_balanco(*raiz*)

*# Caso Esquerda-Esquerda*

*if* balanco > 1 and *nome* < *raiz*.esquerda.nome:

*return* *self*.\_rotacao\_direita(*raiz*)

*# Caso Direita-Direita*

*if* balanco < -1 and *nome* > *raiz*.direita.nome:

*return* *self*.\_rotacao\_esquerda(*raiz*)

*# Caso Esquerda-Direita*

*if* balanco > 1 and *nome* > *raiz*.esquerda.nome:

*raiz*.esquerda = *self*.\_rotacao\_esquerda(*raiz*.esquerda)

*return* *self*.\_rotacao\_direita(*raiz*)

*# Caso Direita-Esquerda*

*if* balanco < -1 and *nome* < *raiz*.direita.nome:

*raiz*.direita = *self*.\_rotacao\_direita(*raiz*.direita)

*return* *self*.\_rotacao\_esquerda(*raiz*)

*return* *raiz*

    def **remover**(*self*, *raiz*, *nome*):

        """Remove um Pokémon da árvore pelo nome e a balanceia."""

*# 1. Remoção padrão de uma Árvore de Busca Binária*

*if* not *raiz*:

*return* *raiz*

*if* *nome* < *raiz*.nome:

*raiz*.esquerda = *self*.remover(*raiz*.esquerda, *nome*)

*elif* *nome* > *raiz*.nome:

*raiz*.direita = *self*.remover(*raiz*.direita, *nome*)

*else*:

*if* *raiz*.esquerda is None:

                temp = *raiz*.direita

*raiz* = None

*return* temp

*elif* *raiz*.direita is None:

                temp = *raiz*.esquerda

*raiz* = None

*return* temp

            temp = *self*.get\_min\_valor\_no(*raiz*.direita)

*raiz*.nome = temp.nome

*raiz*.forca = temp.forca

*raiz*.direita = *self*.remover(*raiz*.direita, temp.nome)

*if* *raiz* is None:

*return* *raiz*

*# 2. Atualiza a altura e rebalanceia a árvore*

*raiz*.altura = 1 + max(*self*.get\_altura(*raiz*.esquerda), *self*.get\_altura(*raiz*.direita))

        balanco = *self*.get\_balanco(*raiz*)

*# Casos de rebalanceamento (semelhantes à inserção)*

*# Caso Esquerda-Esquerda*

*if* balanco > 1 and *self*.get\_balanco(*raiz*.esquerda) >= 0:

*return* *self*.\_rotacao\_direita(*raiz*)

*# Caso Esquerda-Direita*

*if* balanco > 1 and *self*.get\_balanco(*raiz*.esquerda) < 0:

*raiz*.esquerda = *self*.\_rotacao\_esquerda(*raiz*.esquerda)

*return* *self*.\_rotacao\_direita(*raiz*)

*# Caso Direita-Direita*

*if* balanco < -1 and *self*.get\_balanco(*raiz*.direita) <= 0:

*return* *self*.\_rotacao\_esquerda(*raiz*)

*# Caso Direita-Esquerda*

*if* balanco < -1 and *self*.get\_balanco(*raiz*.direita) > 0:

*raiz*.direita = *self*.\_rotacao\_direita(*raiz*.direita)

*return* *self*.\_rotacao\_esquerda(*raiz*)

*return* *raiz*

    def **buscar**(*self*, *raiz*, *nome*):

        """Busca um Pokémon pelo nome e retorna o nó se encontrado."""

*if* *raiz* is None or *raiz*.nome == *nome*:

*return* *raiz*

*if* *nome* < *raiz*.nome:

*return* *self*.buscar(*raiz*.esquerda, *nome*)

*else*:

*return* *self*.buscar(*raiz*.direita, *nome*)

    def **listar\_em\_ordem\_decrescente\_forca**(*self*, *raiz*):

        """

        Retorna uma lista de todos os Pokémon ordenados pela força, do maior para o menor.

        A árvore é ordenada por nome, então precisamos coletar todos e depois ordenar.

        """

        pokemons = []

*self*.\_coletar\_todos\_pokemons(*raiz*, pokemons)

*# Ordena a lista de pokémons pela força em ordem decrescente*

        pokemons.sort(*key*=lambda *p*: *p*.forca, *reverse*=True)

*return* pokemons

    def **\_coletar\_todos\_pokemons**(*self*, *no*, *lista*):

        """Função auxiliar recursiva para percorrer a árvore e coletar todos os nós."""

*if* *no*:

*self*.\_coletar\_todos\_pokemons(*no*.esquerda, *lista*)

*lista*.append(*no*)

*self*.\_coletar\_todos\_pokemons(*no*.direita, *lista*)

    def **imprimir\_arvore**(*self*, *no*, *nivel*=0, *prefixo*="Raiz:"):

        """Imprime a estrutura da árvore no console."""

*if* *no* is not None:

            print(" " \* (*nivel* \* 4) + *prefixo* + *no*.nome + f" (Força: {*no*.forca}, Altura: {*no*.altura})")

*if* *no*.esquerda is not None or *no*.direita is not None:

*self*.imprimir\_arvore(*no*.esquerda, *nivel* + 1, "E---")

*self*.imprimir\_arvore(*no*.direita, *nivel* + 1, "D---")

*# --- Programa Principal e Testes ---*

*if* \_\_name\_\_ == "\_\_main\_\_":

    pokedex\_avl = AVLTree()

    raiz = None

*# Inserindo Pokémon*

    pokemons\_para\_inserir = [

        ("Pikachu", 90), ("Charizard", 120), ("Bulbasaur", 80),

        ("Squirtle", 85), ("Mewtwo", 150), ("Gengar", 110),

        ("Alakazam", 130)

    ]

    print("--- INSERINDO POKÉMON ---")

*for* nome, forca *in* pokemons\_para\_inserir:

        print(f"\nInserindo {nome}...")

        raiz = pokedex\_avl.inserir(raiz, nome, forca)

        pokedex\_avl.imprimir\_arvore(raiz)

        print("-" \* 20)

    print("\n\n--- ESTRUTURA FINAL DA ÁRVORE ---")

    pokedex\_avl.imprimir\_arvore(raiz)

*# Teste de Busca*

    print("\n\n--- TESTANDO BUSCA POR NOME ---")

    nome\_busca = "Mewtwo"

    pokemon\_encontrado = pokedex\_avl.buscar(raiz, nome\_busca)

*if* pokemon\_encontrado:

        print(f"Pokémon '{nome\_busca}' encontrado! Força: {pokemon\_encontrado.forca}")

*else*:

        print(f"Pokémon '{nome\_busca}' não encontrado.")

    nome\_busca = "Ditto"

    pokemon\_encontrado = pokedex\_avl.buscar(raiz, nome\_busca)

*if* pokemon\_encontrado:

        print(f"Pokémon '{nome\_busca}' encontrado! Força: {pokemon\_encontrado.forca}")

*else*:

        print(f"Pokémon '{nome\_busca}' não encontrado.")

*# Teste de Listagem por Força*

    print("\n\n--- LISTANDO POKÉMON POR ORDEM DECRESCENTE DE FORÇA ---")

    lista\_fortes = pokedex\_avl.listar\_em\_ordem\_decrescente\_forca(raiz)

*for* pokemon *in* lista\_fortes:

        print(f"- {pokemon.nome} (Força: {pokemon.forca})")

*# Teste de Remoção*

    print("\n\n--- TESTANDO REMOÇÃO ---")

    nome\_remover = "Bulbasaur"

    print(f"\nRemovendo '{nome\_remover}'...")

    raiz = pokedex\_avl.remover(raiz, nome\_remover)

    pokedex\_avl.imprimir\_arvore(raiz)

    nome\_remover = "Charizard"

    print(f"\nRemovendo '{nome\_remover}'...")

    raiz = pokedex\_avl.remover(raiz, nome\_remover)

    pokedex\_avl.imprimir\_arvore(raiz)

RESULTADOS

Aqui você vai escrever o que aconteceu na sua atividade, que conclusão chegou com o código (ou diagrama ou outros);

# CONCLUSÃO

Em poucas palavras descrever a conclusão da sua atividade

###### 5 - REFERÊNCIAS

Aqui vai colocar as referências bibliográficas.