

Tecnólogo Ciência de Dados

Guilherme Giacomini Teixeira

Analise de dados estruturados:

Trabalho de Avaliação da Unidade 4 da Disciplina Estrutura de Dados

Balneário Camboriú – SC

2025

Balneário Camboriú - SC

2025

Guilherme Giacomini Teixeira

Analise de dados estruturados:

Trabalho de Avaliação da Unidade 4 da Disciplina Estrutura de Dados

­­­­

Trabalho de avaliação da unidade 4 da disciplina Estrutura de Dados apresentado como requisito parcial para a obtenção da média no curso Ciência de Dados.

Professora: Vanessa Matias Leite  
Tutor: João Henrique Correia dos Santos

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 de **Estrutura de Dados**, realizada na **Unidade 4, Aula 2** do curso de **Ciência de Dados**.

O objetivo da atividade foi aprender a realizar uma implementação para a análise de dados estruturados utilizando a linguagem de programação Python. A tarefa proposta consiste em modelar uma rede social como um grafo usando a biblioteca NetworkX , além de implementar uma classe para gerenciar usuários e suas conexões.

O projeto também envolve a utilização de algoritmos de detecção de comunidades e o cálculo de métricas de centralidade para entender a influência dos usuários na rede. Para o desenvolvimento, será utilizada a ferramenta **VSCode**, que permite a execução de código Python de forma eficiente.

Neste relatório, serão descritos os passos seguidos para o desenvolvimento do projeto, detalhando a criação das classes, a implementação dos métodos solicitados e a análise dos resultados obtidos.

# DESENVOLVIMENTO

A linguagem Python é amplamente utilizada em Ciência de Dados por sua sintaxe clara e legível. É uma linguagem de alto nível, interpretada e multiparadigma, com um vasto ecossistema de bibliotecas. Para esta atividade, o Python foi escolhido por sua capacidade de lidar com estruturas de dados complexas, como os grafos, e por sua eficiência na análise de redes.

O projeto foi implementado utilizando o VSCode para criar um programa que modela uma rede social como um grafo. A solução inclui as seguintes classes e funcionalidades:

Classe SocialNetwork: Gerencia o grafo da rede social, armazenando usuários e suas conexões. Possui métodos para adicionar e remover usuários, conectar e desconectar conexões, além de encontrar comunidades e calcular a centralidade dos usuários.

Classe User: Representa um usuário individual com um identificador único (user\_id) e dados associados.

A atividade também exige um script de teste para demonstrar a funcionalidade das classes, realizando operações como a adição de usuários, a criação de conexões e a análise de métricas da rede.

A seguir, o código no VSCode com a implementação do projeto:

import networkx as nx

# Classe para representar um usuário

class User:

def \_\_init\_\_(self, user\_id, data=None):

self.user\_id = user\_id

self.data = data if data is not None else {}

# Método para obter o ID do usuário

def get\_user\_id(self):

return self.user\_id

# Método para atualizar os dados do usuário

def set\_data(self, data):

self.data.update(data)

# Classe para a rede social

class SocialNetwork:

def \_\_init\_\_(self):

# Inicializa um grafo não direcionado da biblioteca NetworkX

self.graph = nx.Graph()

# Adiciona um usuário ao grafo

def add\_user(self, user\_id, user\_data=None):

if not self.graph.has\_node(user\_id):

self.graph.add\_node(user\_id, data=user\_data)

print(f"Usuário {user\_id} adicionado.")

else:

print(f"Usuário {user\_id} já existe.")

# Remove um usuário e suas conexões do grafo

def remove\_user(self, user\_id):

if self.graph.has\_node(user\_id):

self.graph.remove\_node(user\_id)

print(f"Usuário {user\_id} removido.")

else:

print(f"Usuário {user\_id} não encontrado.")

# Conecta dois usuários

def connect\_users(self, user1\_id, user2\_id):

if self.graph.has\_node(user1\_id) and self.graph.has\_node(user2\_id):

if not self.graph.has\_edge(user1\_id, user2\_id):

self.graph.add\_edge(user1\_id, user2\_id)

print(f"Usuários {user1\_id} e {user2\_id} conectados.")

else:

print(f"Usuários {user1\_id} e {user2\_id} já estão conectados.")

else:

print("Um ou ambos os usuários não existem.")

# Desconecta dois usuários

def disconnect\_users(self, user1\_id, user2\_id):

if self.graph.has\_edge(user1\_id, user2\_id):

self.graph.remove\_edge(user1\_id, user2\_id)

print(f"Usuários {user1\_id} e {user2\_id} desconectados.")

else:

print(f"Conexão entre {user1\_id} e {user2\_id} não existe.")

# Encontra comunidades no grafo

def find\_communities(self):

# Utiliza o algoritmo de Louvain para detecção de comunidades

communities = nx.algorithms.community.louvain\_communities(self.graph, seed=42)

print("Comunidades encontradas:")

for i, comm in enumerate(communities):

print(f"Comunidade {i+1}: {comm}")

return communities

# Calcula a centralidade dos usuários

def user\_centralities(self, method='degree'):

if method == 'degree':

centrality = nx.degree\_centrality(self.graph)

elif method == 'betweenness':

centrality = nx.betweenness\_centrality(self.graph)

elif method == 'closeness':

centrality = nx.closeness\_centrality(self.graph)

else:

print("Método de centralidade inválido.")

return {}

print(f"Centralidade ({method}) dos usuários:")

for user, score in centrality.items():

print(f"Usuário {user}: {score:.4f}")

return centrality

# Analisa um subgrafo

def analyze\_subgraph(self, user\_ids):

if all(self.graph.has\_node(uid) for uid in user\_ids):

subgraph = self.graph.subgraph(user\_ids)

print(f"Análise do subgrafo com os usuários {user\_ids}:")

print(f"Número de nós: {subgraph.number\_of\_nodes()}")

print(f"Número de arestas: {subgraph.number\_of\_edges()}")

return subgraph

else:

print("Um ou mais usuários não existem no grafo principal.")

return None

# Script de teste para demonstrar a funcionalidade

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

social\_net = SocialNetwork()

# Adicionando usuários

social\_net.add\_user("Alice", {"name": "Alice Silva", "interests": "Python, Data Science"})

social\_net.add\_user("Bob", {"name": "Bob Souza", "interests": "Machine Learning, AI"})

social\_net.add\_user("Charlie", {"name": "Charlie Santos", "interests": "Web Development"})

social\_net.add\_user("David", {"name": "David Costa", "interests": "Data Science, Statistics"})

social\_net.add\_user("Eve", {"name": "Eve Ferreira", "interests": "Python, Algorithms"})

social\_net.add\_user("Frank", {"name": "Frank Lima", "interests": "Data Visualization"})

print("-" \* 20)

# Conectando usuários

social\_net.connect\_users("Alice", "Bob")

social\_net.connect\_users("Alice", "David")

social\_net.connect\_users("Bob", "David")

social\_net.connect\_users("Bob", "Eve")

social\_net.connect\_users("Charlie", "Frank")

social\_net.connect\_users("David", "Eve")

social\_net.connect\_users("Alice", "Eve")

print("-" \* 20)

# Identificando comunidades

social\_net.find\_communities()

print("-" \* 20)

# Calculando centralidades

social\_net.user\_centralities(method='degree')

social\_net.user\_centralities(method='betweenness')

social\_net.user\_centralities(method='closeness')

print("-" \* 20)

# Analisando subgrafo

social\_net.analyze\_subgraph(["Alice", "Bob", "David"])

print("-" \* 20)

# Removendo um usuário e desconectando usuários

social\_net.remove\_user("Frank")

social\_net.disconnect\_users("Alice", "Eve")

# RESULTADOS:

Nesta atividade, o projeto foi implementado com sucesso utilizando o VSCode e a linguagem Python, seguindo a estrutura de classes proposta. A solução demonstrou de forma prática como as estruturas de dados de grafos podem ser aplicadas para modelar e analisar redes sociais.

Através do código, foi possível simular a adição e remoção de usuários e suas conexões, o que é fundamental para a representação dinâmica de uma rede. A utilização da biblioteca NetworkX simplificou a manipulação do grafo, permitindo a aplicação de algoritmos complexos.

A análise do código-fonte e dos resultados obtidos no script de teste mostrou que as classes SocialNetwork e User foram capazes de gerenciar eficientemente os dados e as interações.

Os algoritmos de detecção de comunidades, como o de Louvain, permitiram identificar subgrupos de usuários com interesses ou conexões em comum. O cálculo de métricas de centralidade (grau, intermediação, proximidade) revelou a influência de cada usuário na rede, indicando quais seriam os nós mais importantes ou influentes.

O projeto demonstrou a importância de ferramentas como a biblioteca NetworkX e a linguagem Python para a Ciência de Dados, ao fornecer insights valiosos sobre a estrutura e o comportamento de uma rede social.

# CONCLUSÃO

A atividade prática proporcionou a oportunidade de aplicar os conceitos teóricos de análise de dados estruturados e grafos, aprendidos na disciplina de Estrutura de Dados. A implementação de um sistema de gerenciamento de rede social utilizando a biblioteca NetworkX e a linguagem Python no ambiente de desenvolvimento VSCode demonstrou a importância dessas ferramentas para a modelagem e análise de interações complexas.

O projeto foi bem-sucedido, com a criação de um código funcional que atendeu a todos os requisitos propostos. A implementação das classes SocialNetwork e User, juntamente com os métodos para adicionar, remover e conectar usuários, foi fundamental para simular o comportamento de uma rede social. A utilização de algoritmos de detecção de comunidades e o cálculo de métricas de centralidade confirmaram a eficácia dessa abordagem para extrair insights valiosos sobre a estrutura da rede e a influência de seus usuários.

Em conclusão, este trabalho reforçou o aprendizado sobre a aplicação de estruturas de dados avançadas, como grafos, em problemas práticos. Ele mostrou como a escolha de ferramentas e bibliotecas eficientes é crucial para o desenvolvimento de softwares robustos e para a análise de dados complexos de forma eficaz.

.

# REFERÊNCIAS

ALVES, W. P. **Programação Python**: aprenda de forma rápida. São Paulo: Expressa, 2021.

DIAS, M. A. **Análise de dados estruturados**. Estrutura de Dados, 2021. Disponível em: https://bit.ly/49g36r9. Acesso em: 17 fev. 2024.

LAMBERT, K. A.**Fundamentos de Python**: estruturas de dados. São Paulo: Cengage Learning, 2022.

SZWARCFITER, J. L.; MARKENZON, L. **Estruturas de dados e seus algoritmo**s. 3. ed. Rio de Janeiro: LTC, 2020.

LEITE, V. M. **Análise de dados estruturados**. Aula 2. In: UNOPAR ANHANGUERA. Estrutura de Dados. [Material de curso]. Unidade 4, Estruturas de Dados Avançados e Análise de Dados. [S. l.]: Anhanguera Unopar, 2025. Acesso restrito.