Projeto I de Estrutura de dados: Rede social simplificada

Universidade Politécnica de Pernambuco

Prof. Dr. Cleyton Mario Oliveira da Silva

Aluno: Diego Augusto Soares Amaral

09/10/2025

Resumo: Neste relatório foi descrita a resolução do projeto, onde foi necessário desenvolver uma aplicação que utilize TADs e listas como base, utilizar pelo menos uma estrutura de dados(fila, Deque e etc), e também um mecanismo de busca e ordenação otimizado. Foi escolhido fazer uma rede social simplificada, onde foi utilizada uma lista para armazenar os usuários cadastrados, um deque para o feed de posts universal, um merged sort como mecanismo de ordenação otimizado para ordenar os posts por número de likes, e por fim uma busca binária para buscar os usuários por ID.
Palavras - chave: Estrutura de dados, TADs, Mecanismo de ordenação.

Abstract: This report describes the project's resolution, which required developing an application that uses ADTs and lists as a basis, utilizing at least one data structure (queue, deque, etc.), and also an optimized search and sorting mechanism. The chosen approach was to create a simplified social network, using a list to store registered users, a deque for the universal post feed, a merged sort as an optimized sorting mechanism to sort posts by number of likes, and finally, a binary search to search for users by ID.
KEYWORDS: Data structure. ADTs. Sorting mechanism.

Conteúdo

1 Introdução

1.1 Apresentação do problema

O projeto tem como objetivo utilizar e demonstrar o funcionamento de estruturas de dados diversas, assim como um mecanismo de busca ou ordenação otimizado. Ele vai além de um simples código que utilizaria a estrutura escolhida e rodaria alguns comandos. Ele cria relações profundas e complexas que vão de encontro com temas e situações da vida real, onde não é perceptível para todos, mas a maioria dos grandes programas utilizam estruturas de dados e mecanismos de busca e ordenação otimizados como base para sua estrutura.

1.2 Tema escolhido

O tema escolhido foi uma rede social simplificada. Ele consegue explorar e demonstrar de um modo prático e eficiente como essas estruturas e mecanismos funcionam na vida real, como por exemplo o mecanismo de ordenação que consegue de maneira rápida e eficiente ordenar todos os posts por like, ou o formato de deque que permite manipular e simular um sistema de notificações, onde a notificação mais recente é adicionada na esquerda para ser a primeira a ser vista.

2 Objetivos

Os objetivos do projeto são:

1. Modelar e implementar um simulador de rede social, utilizando os conceitos de tipos abstratos de dados(TADs).
2. Utilizar Listas, Pilhas e Deque para gerenciar usuários, ações e notificações respectivamente.
3. Implementar um algoritmo de ordenação eficiente, Merge Sort para adicionar uma funcionalidade de verificar o feed em alta, e um algoritmo de busca eficiente que foi a busca binária.
4. Desenvolver um sistema interativo via terminal para o usuário.

3 Desenvolvimento

3.1 Modelagem do sistema e TADs

O sistema se baseia em três classes: classe Usuario, classe Post e classe RedeSocial. A classe Usuario cria um objeto do tipo usuário, que possui um id próprio e único gerado pelo próprio código. É pedido que se forneça um nome e um email para realizar o cadastro, e é o nome que aparece no feed e em outras funções aparentes do sistema, sendo o id só utilizado para busca inteligente.

A classe Post cria um objeto do tipo post, onde também vai ter um id único criado pelo próprio sistema, e vai ser necessário passar como parâmetro o texto do post, e o autor que está escrevendo o post.

Por fim a classe rede social, que é a classe gestora, onde se utiliza composição para fazer as relações entre as 3 classes, que seria a relação "tem um", onde na rede social "tem um"usuário e "tem um"post por assim dizer. Ela também é onde todas as funções principais vão rodar, como a função de adicionar uma curtida em um post, onde é passado o id do post e o id do usuário que está curtindo o post, e uma notificação de curtida é enviada para o autor do post.

3.2 Descrição das ferramentas

3.2.1 Lista

Listas são conjuntos de dados onde existe um encadeamento sequencial dos elementos. Elementos podem ser inseridos ou removidos a depender da ordem do conjunto. Foi utilizada uma lista para salvar os usuários pois a ordem de inserção não é crítica, e o acesso principal se dá pela busca de um elemento.

3.2.2 Pilha

Uma pilha é uma estrutura de dados linear que segue a ordem LIFO(Last in, first out), onde o último a entrar é o primeiro a sair. A natureza da pilha (LIFO) modela perfeitamente a função desfazer, onde todas as ações feitas pelo usuário são salvas na pilha própria, e quando é pedido para desfazer a última ação, a que foi feita por último é desfeita, ou seja, a última a entrar na pilha.

3.2.3 Deque (Double-Ended Queue)

A estrutura de dados Deque, que traduzindo seria fila de duas pontas, é como uma lista simples, mas que permite remoção e inserção tanto no início quanto no final. Foi utilizado para gerenciar as notificações dos usuários, pois permite inserção direta na esquerda com $appendleft()$(O(1)), e mantém a ordem cronológica inversa de forma natural, sendo ideal para manter a performance constante da função de adicionar posts.

3.3 Algoritmos implementados

3.3.1 Algoritmo de ordenação: Mergesort

O algoritmo de mergesort utiliza o paradigma de dividir para conquistar, no qual ele divide a lista pela metade e separa em esquerda e direita, e assim ordena a lista pela quantidade de likes para a função de "ver em alta". Foi escolhido ele ao invés do Quicksort por exemplo, porque no pior dos casos o Quicksort é (O(n^2)), enquanto que o Mergesort tem um desempenho constante de (O($nlogn$)). Também no Mergesort foi definido um critério de desempate, onde ele vai analisar uma tupla, que possui o número de likes do post como primeiro critério, e depois o ID do post. Se os posts têm o mesmo número de likes, ele compara o id do post para saber qual foi mais recente; o maior id é do post mais recente, então sempre vai ficar ordenado primeiramente no maior número de likes, e no caso de empate aparece primeiro o mais recente.

3.3.2 Algoritmo de busca: Busca linear e busca binária

Para o projeto, foi utilizado um algoritmo de busca simples, já que ele busca o Deque de feed global que é onde fica salvo cada post. Para esse problema, uma busca simples já foi um mecanismo suficiente. Já para percorrer a lista de usuários e procurar por ID, foi utilizada a busca binária, pois como a lista de usuários está sempre ordenada em ordem crescente do ID, esse sistema de busca é perfeito para o programa.

4 Resultado e discussão

Foi utilizado o ambiente do Colab para esse projeto, pois, como não era necessária interface gráfica, ele é mais leve e eficiente que algumas IDEs como o VS Code, o que dava flexibilidade para trabalhar no código independente do local presencial. A seguir vão ter algumas screenshots para mostrar a estrutura e que o programa funciona de maneira correta e eficiente.

Fonte: Diego Amaral (2025)

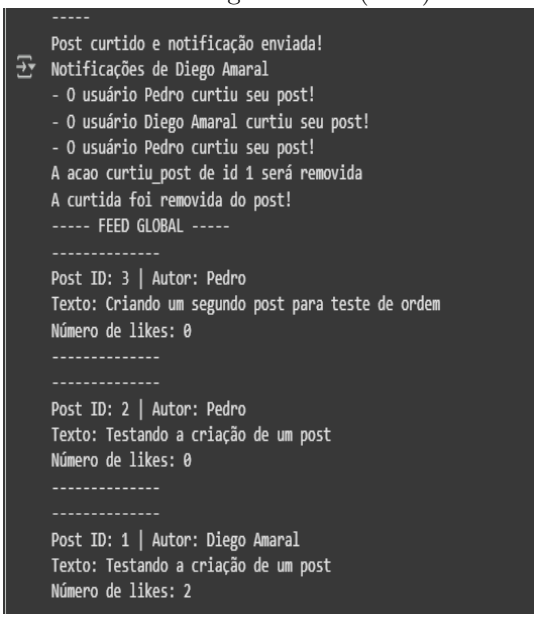

Figura 1: Sistema de notificações e feed global organizado pelo mais recente primeiro.

Fonte: Diego Amaral (2025)

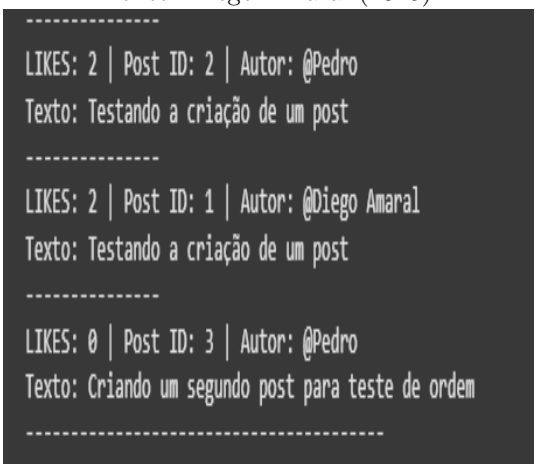

Figura 2: Ordenação por número de likes e em caso de empate, o mais recente aparece primeiro

5 Conclusão

Os objetivos do projeto foram alcançados com sucesso, onde foi possível demonstrar com um contexto da vida real, a utilização na prática das estruturas de dados vistas em sala, assim como a utilização de mecanismos de ordenação e busca eficientes. O projeto ajudou a solidificar o conhecimento em ambos os assuntos, onde é praticando que realmente se aprende, e utilizando uma prática ainda mais avançada e contextualizada, facilita ainda mais a memorização e assimilação do conteúdo. O projeto possui algumas limitações, como ainda não é possível remover um usuário e ainda não é possível fazer uma conta e entrar nela para operar individualmente dentro do código.

As possíveis melhorias futuras seriam a implementação de grafos para melhorar e trazer ainda mais relações entre os usuários, uma interface gráfica para ficar ainda mais parecido com uma rede social, também uma tela de login e senha para um usuário cadastrar interagir pela própria conta do jeito que ele quiser e por fim, a função de remover usuário. Para transformar este protótipo em uma aplicação persistente, onde os dados são mantidos entre diferentes execuções, o próximo passo seria implementar um sistema de serialização. A abordagem mais direta em Python seria utilizar a biblioteca pickle para salvar o estado do objeto RedeSocial em um arquivo e carregá-lo no início de uma nova sessão.

6 Referências

1. Introduction to algorithms / Thomas H. Cormen . . . [et al.].—3rd ed.

2. SILVA, Cleyton Mario Oliveira da. [Aula 06]. [Slides de aula]. Disciplina de Algoritmo e Estrutura de Dados, Universidade Politécnica de Pernambuco, 2025.