RELATÓRIO: HIERARQUIA DE CLASSES

Curso: Tópicos em Ciência de Dados A (MAE013)

Professor: Milton R Ramirez

Integrantes

Leonardo Domingos (DRE: 120168324);

Hierarquia de Classes

1. Introdução

Nesse projeto, iremos implementar as seguintes classes:

1. **Array**
2. **Linked List**
3. **Doubly Linked List**
4. **Circular Linked List**
5. **Queue**
6. **Stack**

Também iremos usar essas classes para abordar alguns tópicos secundários, como a Lista de Prioridade, ou calcular a interseção entre linhas num formato geométrico.

2. Metodologia

O projeto foi desenvolvido apenas com Python e suas bibliotecas padrão. Ele consiste em um arquivo para cada classe implementada, e um arquivo central **final.py,** que guarda as funções de teste de cada uma dessas classes. Assim, cada classe tem o seguinte par:

- **Arquivo.py**
- Função de teste no **final.py**

A classe mais importante é a **LinearDataStructure**, classe pai de quase todas as outras classes. Outra classe importante é a **LinkedList** (classe filha da LinearDataStructure) que é responsável por generalizar o comportamento de todas as listas a serem implementadas.

3. Comentários sobre o Desenvolvimento

3.1. Comentários Gerais

Primeiramente, o tamanho desse projeto, devido a quantidade de classes a serem implementadas, realmente testam a sua capacidade de desenvolver código legível, escalável e inteligível. No início, comecei o desenvolvimento da mesma forma que dos outros projetos: tentando fazer tudo em um arquivo só. Essa abordagem rapidamente se provou inadequada, uma vez que o arquivo começou a ter centenas de linhas de código que não se relacionavam umas com as outras (não faz sentido ter uma classe Array e LinkedList no mesmo arquivo, por exemplo), o que não só deixava o código mais confuso, como também dificultava encontrar as partes do código que eu estava interessado.

Imediatamente após dividir em arquivos diferentes, consegui fluir bem mais rapidamente. Também tirei o tempo para fazer a implementação dos métodos das classes da forma correta, com o comentário explicando o que aquele método fazia. Isso foi importante, já que no VSCode, ao passar o mouse por cima de algum método ou função, justamente esse comentário será mostrado. E é bem mais fácil e rápido desenvolver em cima de código que se fez no dia anterior se ele está bem explicado e organizado.

3.2. Comentários das Implementações

Ao longo de praticamente todo o projeto tive o mesmo problema comum entre as classes: como lidar com colocar/remover itens que estão no meio da estrutura de dados. A reorganização sempre me dava certo trabalho de entender e conseguir implementar da maneira correta, principalmente na CircularLinkedList. Conceitualmente é fácil de entender, mas implementar era outra história.

Falando em Lista, essas foram certamente as mais difíceis de implementar. Lidar com a head e tail não é algo imediatamente claro, e erros no código raramente vão evitar o seu programa de compilar (só se errar muito). De novo, em especial a CircularLinkedList, que na minha opinião é a mais difícil de se implementar, no nível de que eu estava tendo problemas em simplesmente printar os elementos nela sem cair num loop infinito. Eventualmente “resolvi” isso, ao transformar a váriavel numa lista (isso pode ser visto na função de teste test_circular_linked_list() em final.py. toda vez que preciso printar a lista, eu faço list(lista)).

Por fim, a implemetação da FiguraGeometrica.py, classe que resolve o exercício da CircularLinkedList , também foi complexa. Não só devido ao fato de que usar essa classe realmente expôs inúmeras falhas na minha CircularLinkedList, como também porque as contas para fazer a detecção dos cenários não são fáceis de se deduzir. Felizmente, esse é um problema comum, que tem várias explicações na internet.

Em resumo, a implementação da CircularLinkedList e FiguraGeometrica tomaram por volta de 50% do tempo de desenvolvimento.