

Contribuições

GITHUB: <https://github.com/ArthurBusquet/trabalho-teoria-dos-grafos>

Arthur: responsável por definir e construir toda a estrutura do armazenamento em lista encadeada, ou seja, criar as classes `VerticeEncadeado`, `ArestaEncadeada`, `GrafoLista` e `Lista Encadeada`, além de definir as responsabilidades, e criar os métodos e propriedades de cada uma. Dentro da representação por lista encadeada, criei todos os métodos e propriedades utilizados neste primeiro trabalho, dentro dos métodos da classe `GrafoLista`, eu destaco `get_vertice`, `set_vertice`, `get_aresta`, `set_aresta` e `get_vizinhos` que são fundamentais para a manipulação do grafo em outros métodos. Além dos métodos genéricos da lista encadeada, que funciona tanto para armazenar vértice quanto arestas. Outro ponto que gostaria de destacar diz respeito ao tratamento das conexões de um vértice qualquer, no qual criei métodos que facilitam a remoção e adição de novas arestas na lista de conexões, ao mesmo tempo que registro e atualiza automaticamente o grau de cada vértice.

No grafo matriz criei os métodos `set_aresta` e `set_vertice` a fim de facilitar a leitura do arquivo `grafo.txt` e a criação do grafo.

Ajudei a garantir que o conceito de herança estava sendo bem aplicado, uma vez que criei os métodos genéricos `eh_completo`, `get_grau` e `get_ordem` apenas na classe pai.

Também criei o método `carrega_grafo` genérico que faz a leitura do `grafo.txt` na classe pai e se utiliza dos métodos auxiliares `set_vertice` e `set_aresta`, aplicando o conceito de polimorfia.

Por fim, gerenciei o novo repositório no github, resolvendo conflitos em merges, avaliando pull requests e apoiando os colegas em eventuais dúvidas sobre git.

Bernardo: Responsável pela implementação e manutenção das funções `eh_arvore()` e `possui_articulacao()` na estrutura de dados `Grafo Lista`, e da função `possui_ponte()` na estrutura de dados `Grafo Matriz`. Além disso, desempenha um papel crucial na gestão do repositório de código, executando merges de branches e supervisionando a criação e resolução de conflitos entre branches, garantindo assim a integridade e colaboração no desenvolvimento do projeto.

Geovanni: Responsável pela implementação das funções "Possui Ponte", "Possui Articulação" e "Novo Grafo".

- A função "Possui Ponte" utiliza uma estrutura de dados do tipo `Lista` para identificar se o grafo possui pontes.
- A função "Possui Articulação" utiliza uma estrutura de dados do tipo `Matriz` para verificar a existência de pontos de articulação no grafo.
- A função "Novo Grafo" utiliza uma estrutura de dados do tipo `Lista` para criar um novo grafo a partir de um grafo existente.

Pedro: Responsável pela elaboração da documentação detalhada da primeira fase do trabalho, incluindo a descrição dos objetivos, metodologia e resultados alcançados. Além

disso, implementou com sucesso a integração de um novo tipo de nó e uma nova aresta em ambos os modelos de grafos utilizados na segunda fase do projeto, contribuindo para a expansão e aprimoramento da funcionalidade do sistema..

Jorge: Foi responsável por estruturar toda a representação do grafo utilizando matriz de adjacência, criando a classe GrafoMatriz e implementando seus principais métodos. Dentro dessa estrutura, desenvolvi os métodos `get_vertice`, `set_vertice`, `get_aresta`, `set_aresta` e `get_vizinhos`, fundamentais para a manipulação do grafo e sua utilização em outros métodos. Além disso, implementei os métodos `get_grau` e `n_conexo` para análise da conectividade do grafo e participei da estruturação da hierarquia, garantindo a correta aplicação da herança. Também contribui para a otimização das operações de inserção e remoção de arestas e vértices, garantindo eficiência na manipulação da matriz.