Framework de Atividade de Grafo: Framework de Grafos

Pedro Henrique Damann

[1] Estudante de Graduação em Bacharelado em Ciências da Computação, IFC - *Campus* Rio do Sul. E-mail: pedroo-phd@outlook.com
[2] Orientador, Professor Rodrigo Curvello EBTT, IFC - *Campus* Rio do Sul. E-mail: rodrigo.curvello@ifc.edu.br

RESUMO

Este artigo apresenta um framework para atividades relacionadas a grafos, com foco específico na implementação e uso dos algoritmos BFS, DFS e Dijkstra. O framework foi desenvolvido para permitir que os usuários visualizem estruturas de grafos e explorem diversas operações algorítmicas de maneira simples e interativa. Esta ferramenta tem como objetivo facilitar a compreensão dos conceitos de teoria dos grafos, proporcionando uma aplicação prática que conecta os conceitos teóricos às representações visuais.

Palavras-chave: BFS; DFS; Dijkstra; Ferramenta; Teoria dos Grafos.

INTRODUÇÃO

A teoria dos grafos é uma área fundamental da ciência da computação, tendo seu início com Leonhard Euler em 1936, suas aplicações vão desde redes de computadores até a modelagem de relações sociais e problemas logísticos. No entanto, este assunto tende a ser desafiador para quem se aprofunda nele, logo, este trabalho visa ajudar o entendimento sobre esta área utilizando o framework desenvolvido que possibilita a criação e manipulação de grafos, utilizando os algoritmos clássicos de Busca em Largura (BFS), Busca em Profundidade (DFS) e o algoritmo de Dijkstra para encontrar o caminho mais curto entre os grafos.

O principal objetivo deste framework é proporcionar uma ferramenta interativa que permita aos usuários não só criar e visualizar grafos, mas também entender de maneira prática como esses algoritmos citados operam sobre diferentes tipos de grafos. Espera-se que, por meio desse framework, o usuário possa ter uma visão mais clara e tangível dos conceitos abstratos presentes na teoria dos grafos.

PROCEDIMENTOS METODOLÓGICOS

A construção do framework foi orientada pela necessidade de criar uma ferramenta que fosse tanto prática quanto ilustrativa. Utilizando a linguagem de programação Java, o desenvolvimento seguiu os princípios de design orientado a objetos, com ênfase na modularidade e reusabilidade de código. Foram implementados três algoritmos principais:

ISSN 2318-2342

1. **Busca em Largura (BFS):** Este algoritmo foi escolhido por sua capacidade de explorar grafos de maneira sistemática, sendo útil para encontrar o caminho mais curto em grafos não ponderados.
2. **Busca em Profundidade (DFS):** A DFS foi implementada para permitir a exploração profunda de grafos, útil em aplicações como a detecção de ciclos e a geração de labirintos.
3. **Algoritmo de Dijkstra:** Este algoritmo foi escolhido pela sua eficiência em encontrar o caminho mais curto através dos grafos ponderados, sendo amplamente utilizado em aplicações de roteamento e redes.

A arquitetura do framework é composta por classes principais como “Graph”, “Node”, “Main”, “Aresta” e outras classes auxiliares como “NodeDistancePair”, garantindo flexibilidade para futuras adições. Testes foram conduzidos usando JUnit para assegurar que todas as implementações fossem testadas corretamente.

1. Nodo: Essa classe aceita dados do tipo genérico (<T>) e tem como seu objetivo representar um nodo dentro do grafo, armazenando o valor inserido pelo usuário.
2. Aresta: Essa classe aceita dados do tipo genérico (<T>) e tem como seu propósito representar uma aresta que interliga um nodo entre outro, salvando as informações de ligamento e qual é o peso desta aresta.
3. Graph: Essa classe é a que vai conter os algoritmos citados anteriormente e também os métodos de criação de um novo Node e interligação das arestas.
4. Main: Essa classe é a qual onde os algoritmos para criar o grafo vão ser chamados.
5. NodeDistancePair: Essa classe é auxiliar do algoritmo Dijkstra, com o objetivo de armazenar um Node e sua distância associada.

RESULTADOS E DISCUSSÃO

O framework foi testado em diversos cenários, simulando diferentes tipos de grafos, desde os mais simples até os mais complexos. Os testes mostraram que o algoritmo BFS é eficaz em encontrar caminhos em grafos não ponderados, enquanto o DFS é particularmente útil em cenários onde a profundidade do grafo é uma preocupação maior. O algoritmo de Dijkstra demonstrou eficiência na busca de caminhos mínimos em grafos ponderados, com tempos de execução compatíveis com as expectativas teóricas.

CONSIDERAÇÕES FINAIS

Este trabalho apresentou um framework robusto para a criação e manipulação de grafos, incorporando os algoritmos BFS, DFS e Dijkstra. A ferramenta proporciona aos usuários uma maneira intuitiva de explorar conceitos fundamentais da teoria dos grafos.

ISSN 2318-2342

Futuras expansões do framework podem incluir a implementação de outros algoritmos, como o de Bellman-Ford ou a busca A*, além de aprimorar a interface de visualização para suportar grafos maiores e mais complexos. A integração com outras plataformas educacionais também pode ser uma direção promissora para o aumento do impacto do framework.

REFERÊNCIAS

NAGEL, Vinicio A. B., **Algoritmos de busca em grafos**. 2015. Disponível em: http://200.135.58.21:4000/items/0221c3e2-13f0-49ce-9f91-240721372376. Acesso em: 22 ago. 2024.

HOLANDA, Bruno. **Grafos: conceitos, algoritmos e aplicações.** 2011. Disponível em: https://www.obm.org.br/content/uploads/2017/01/Nivel1_grafos_bruno.pdf. Acesso em: 22 ago. 2024.

HERMUCHE, Anwar. **Métodos de busca em grafos: BFS e DFS**. Medium, 2024. Disponível em: https://medium.com/@anwarhermuche/m%C3%A9todos-de-busca-em-grafos-bfs-dfs-cf17761a0dd9. Acesso em: 22 ago. 2024.

KADAMBALAMATCLO**, Java Program to Implement Adjacency List**. GeeksforGeeks, 2024. Disponível em: https://www.geeksforgeeks.org/java-program-to-implement-adjacency-list/. Acesso em: 22 ago. 2024.

ISSN 2318-2342