Busca de menor caminho em labirintos - aplicação de algoritmo de busca em grafos

Gustavo P. L. Faria[1], Maísa V. Moreira[1], Prof. Felippe G. L. Rodrigues[1]

[1]Curso de Ciência da Computação – Centro Universitário Instituto de Educação Superior de Brasília (IESB) – Brasília, DF – Brasil

gustavo.faria@iesb.edu.br, maisa.moreira@iesb.edu.br, rambim@gmail.com

Abstract. *In this article, the application of graph search algorithms to find the shortest path in unknown mazes, represented as unweighted graphs, is discussed. Using an API that provides information about the vertices and edges of the graph, an uninformed search algorithm is developed to demonstrate the algorithm's effectiveness, validating its applicability and efficiency.*

Resumo. *Neste artigo, é abordada a aplicação de algoritmos de busca em grafos para encontrar o menor caminho em labirintos desconhecidos, representados como grafos não ponderados. Utilizando uma API que fornece informações sobre os vértices e arestas do grafo, é desenvolvido um algoritmo de busca não informada para demonstrar a eficácia do algoritmo, validando sua aplicabilidade e eficiência.*

1. Introdução

Labirintos são estruturas intrincadas que desafiam a capacidade de navegação devido à complexidade de seus caminhos e passagens. O objetivo principal ao lidar com um labirinto é encontrar uma saída ou um destino específico dentro desse espaço, a partir de uma localização inicial.

Esse projeto explora então a resolução de labirintos desconhecidos utilizando como recursos algoritmos de busca em grafos para encontrar o menor caminho até um vértice predeterminado que esteja conectado a esse grafo.

2. Objetivos

4.1. Objetivos gerais

Esse trabalho tem como objetivo explorar os algoritmos de busca em grafos para encontrar o menor caminho em um labirinto desconhecido, adotando um deles para ser desenvolvido na linguagem Java e executado. Os critérios para gerar o labirinto desconhecido serão previamente estabelecidos, de forma que o estudo do algoritmo adotado seja adequado para solucionar o problema proposto.

4.2. Objetivos específicos

Os objetivos específicos desse trabalho são:

- Analisar os algoritmos de teoria dos grafos que podem ser usados para a resolução de labirintos desconhecidos;

- Consumir uma API REST que gere labirintos aleatórios desconhecidos ao algoritmo de busca em grafos;

- Implementar um algoritmo de busca em grafos que encontre o menor caminho até a saída do labirinto na linguagem Java;

- Analisar o desempenho do algoritmo adotado em diferentes tipos de labirintos.

3. Metodologia

Nessa pesquisa será adotada uma abordagem inicial para embasar e contextualizar o objeto de estudo através de trabalhos correlatos, visando os algoritmos de busca em grafos mais utilizados e quais estratégias eles utilizam. Essa etapa visa garantir que a abordagem esteja alinhada com o estado da arte e possa direcionar os esforços de pesquisa para solucionar o problema proposto.

No que diz respeito ao labirinto que deve ser percorrido, uma API externa foi desenvolvida para criar diferentes possibilidades de labirinto. No contexto dessa pesquisa, o labirinto é desconhecido pelo algoritmo. Logo, ao contrário de labirintos predefinidos com estruturas de matriz conhecidas, os labirintos gerados pela API não podem ser representados eficientemente como matrizes estáticas. Essa característica impõe um desafio significativo, pois o labirinto não pode ser mapeado como uma matriz bidimensional de forma direta.

Em relação à implementação do algoritmo e consumo da API, optou-se por utilizar a linguagem de programação Java para desenvolvimento.

Por fim, a avaliação do desempenho do algoritmo será feita através de testes em diferentes labirintos, conduzidos para avaliar a eficácia e a eficiência do algoritmo em encontrar uma saída nos labirintos fornecidos.

Dessa forma, a metodologia adotada incorpora uma revisão da literatura, a escolha fundamentada de uma linguagem de programação, o uso de uma API especializada e uma abordagem de avaliação para garantir a qualidade e a relevância da pesquisa.

4. Referencial teórico

4.1. Grafos

De acordo com Dhali et al. (2010, p. 52), a Teoria dos Grafos emerge como uma ferramenta extremamente útil para desenvolver métodos e técnicas eficazes de resolução de labirintos. Essa teoria oferece um conjunto de conceitos e princípios matemáticos que podem ser aplicados para modelar e encontrar soluções para os desafios que os labirintos apresentam.

Assim, entende-se um grafo por um conjunto finito não vazio vértices (ou nós), e um conjunto finito de arestas, onde o vértice consiste numa extremidade do conjunto e uma aresta é o caminho que conecta dois vértices (BONDY e MURTY, 2008, p. 2). Esse conceito pode ser aplicado então para a resolução de labirintos, onde um ponto dentro dele pode ser interpretado como um vértice, os movimentos dentro dele criam novos vértices e o caminho entre esses pontos são as arestas.

Quando se trata de utilizar recursos computacionais para navegar através de labirintos e encontrar um destino específico dentro deles, foram desenvolvidos algoritmos de busca que empregam uma variedade de estratégias. A intenção desses algoritmos varia desde localizar a saída do labirinto, calcular a rota mais curta para alcançá-la ou até mesmo encontrar a rota mais rápida, dentre outros. Ou seja, eles não apenas procuram a resposta esperada, mas também otimizam o caminho a ser percorrido para chegar lá.

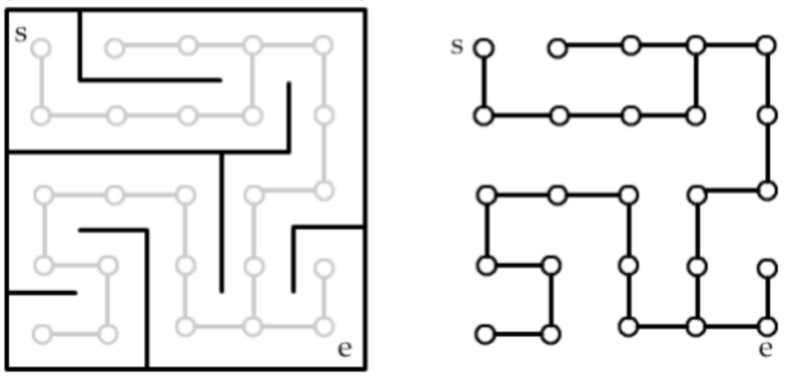

Figura 1. Exemplo de conversão de um labirinto em grafo. Fonte: https://www.cs.umd.edu/class/spring2019/cmsc132-020X-040X/Project8/proj8.html.

O modelo aqui proposto trata de uma busca não informada, ou seja, não há informações adicionais sobre os vértices ou arestas além das próprias informações de conectividade. Isso significa que o algoritmo não possui conhecimento prévio sobre a localização ou a qualidade das soluções no espaço de estados, devendo explorar o grafo de forma sistemática para encontrar a solução.

4.2. API

API é a sigla utilizada para *Application Programming Interface* (Interface de Programação de Aplicação) e trata-se de uma coleção de protocolos e regras que permitem que diferentes aplicativos de software interajam e troquem informações entre si (SHAHI et. al, 2023, p.673).

Segundo Shahi et. al (2023, p. 673), as vantagens de utilizar APIs REST são sua flexibilidade e a capacidade de interagir com outros sistemas e aplicativos com facilidade, além de poderem ser usadas com qualquer linguagem de programação, fornecendo dados e funcionalidades de uma aplicação.

Portanto, a decisão de empregar uma API no projeto foi fundamentada em considerações técnicas relevantes. Optou-se então por implementar uma API que será consumida pelo programa desenvolvido, onde sua principal função é criar os labirintos aleatórios, que serão empregados como cenários para as atividades de navegação e resolução implementadas pelo algoritmo desse projeto.

A API possui três *endpoints* com funcionalidades específicas:

- Iniciar: permite ao usuário iniciar a exploração do labirinto.

- Movimentar: permite ao usuário se mover pelo labirinto.

- Valida Caminho: valida se a sequência de movimentos fornecida é um caminho válido no labirinto.

Como parte essencial do projeto, o algoritmo de busca *Depth-First Search* (DFS - em português: busca em profundidade) é o ponto de partida escolhido para explorar o grafo do labirinto, por ser um dos algoritmos de busca mais empregados em problemas de busca e exploração de grafos. Sua técnica garante que, ao percorrer sistematicamente as arestas de um determinado grafo, cada aresta é percorrida exatamente uma vez e cada vértice é visitado pelo menos uma vez — começando pela raiz do grafo e explorando suas ramificações até a região mais profunda, até ter visitado todos os vértices.

É importante destacar que a busca em profundidade não resolve um problema específico, mas funciona como um pré-processamento para a resolução eficiente de vários problemas concretos, pois ajuda a compreender a forma e as informações sobre um grafo (USP). O DFS garante que a saída será encontrada, já que todos os vértices serão visitados, porém não há indícios que possam afirmar que ele encontrará a saída com o menor caminho possível.

5. Desenvolvimento

Como abordado anteriormente, o DFS foi utilizado para mapear o grafo, explorando sistematicamente as arestas e registrando as informações necessárias, como o vértice de início, vértice destino e todas as ramificações.

Uma vez que o grafo está mapeado, ao buscar o menor caminho possível para achar a saída em um grafo não ponderado com pesos de aresta iguais, o algoritmo *Breadth-First Search* (BFS - em português: busca em largura) foi escolhido como uma das alternativas a serem testadas.

O BFS é uma escolha apropriada para encontrar o menor caminho em labirintos não ponderados devido à sua simplicidade, eficiência em termos de tempo e garantia de encontrar o menor caminho quando todos os pesos das arestas são iguais. Ele é frequentemente usado em aplicações de busca em largura, onde a ênfase está na descoberta do caminho mais curto em termos de movimentos.

Na funcionalidade desenvolvida, três estruturas de dados são utilizadas para realizar a busca do caminho mais curto: a **distância** atribui a cada vértice uma distância mínima a partir do vértice de origem; o **pai** mantém o registro dos pais de cada vértice na árvore de busca do caminho mais curto; e a **fila** é usada para realizar a busca em largura (BFS), determinando a ordem em que os vértices são explorados. Essas estruturas desempenham papéis cruciais: **distância** registra as distâncias mínimas, **pai** rastreia a estrutura do caminho, e **fila** permite a expansão ordenada dos vértices, tornando possível encontrar o caminho mais curto entre a origem e o destino no grafo não ponderado.

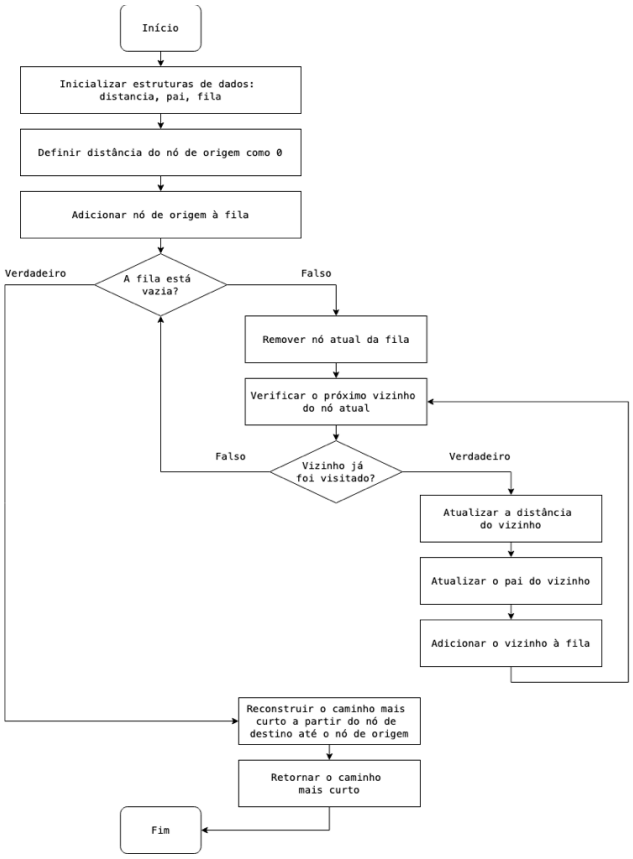

Figura 2. Fluxograma do funcionamento do algoritmo BFS desenvolvido. Fonte: os autores (2023).

Após o desenvolvimento do algoritmo proposto, uma série de testes foram conduzidos, a partir dos labirintos fornecidos pela API instalada localmente.

Tabela 1. Avaliação de desempenho do algoritmo BFS desenvolvido

Identificador do labirinto	Total de vértices	Total de arestas	Início	Destino	Total de movimentos até a saída	Tempo de execução	Achou a saída?	Saída é válida?
Maze sample	10	13	8	9	4	517 ms	Sim	Sim
Maze sample 2	10	14	8	9	3	509 ms	Sim	Sim
Medium maze	40	44	1	22	32	1827 ms	Sim	Sim
Large maze	491	968	161	307	35	18911 ms	Sim	Sim

6. Conclusão

Os resultados obtidos nessa pesquisa refletiram consistentemente a capacidade do algoritmo em encontrar o menor caminho em todos os cenários propostos, validando, assim, sua eficácia.

É importante destacar que, embora o BFS demonstre eficiência em termos de complexidade, o tempo real de execução pode variar de acordo com o tamanho e a estrutura do grafo, bem como a implementação específica do algoritmo. Em grafos de

maior porte ou com maior densidade, é natural que o BFS demande um tempo considerável.

A complexidade O(V + E), onde V representa o número de vértices e E o número de arestas no grafo, fornece uma estimativa do comportamento do algoritmo à medida que o tamanho do grafo aumenta. No entanto, é crucial observar que o tempo de execução real também é influenciado por outros fatores, como o desempenho da unidade central de processamento (CPU) e a gestão de recursos de memória.

Com base nos resultados apresentados neste artigo, é evidente que o tempo necessário para encontrar a saída de um labirinto está diretamente relacionado com sua complexidade. Isso está em consonância com a complexidade linear do algoritmo BFS. Em resumo, o projeto alcançou com sucesso os objetivos propostos, validando a eficiência do algoritmo em encontrar o menor caminho em labirintos não ponderados.

Referências

BONDY, J. A.; MURTY, U. S. R. Graduate Texts in Mathematics: Graph Theory. Ed. 2008. Springer, 2008.

DHALI, M. A.; SYED, A.; FARID, H. M. A. B. A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory. In: Artificial Intelligence and Computational Intelligence (AICI), 2010. DOI: 10.1109/AICI.2010.18.

SHAHI, A.; BARGE A.; BUTALA, A.; PATIL, S. Rest API based Web Interface for Blogging Application. In: 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS).

USP. Algoritmos para Grafos - Aulas: Depth-First Search (DFS). Disponível em: https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dfs.html. Acesso em: 05 out. 2023.