Busca de menor caminho em labirintos desconhecidos - 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 (Sadik et al., 2010, p. 52). 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 que se estendem por mais de uma dimensão, 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

2.1. 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.

2.2. 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 labirintos fornecidos pela API.

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, o que impõe um desafio significativo na adoção de uma estratégia de mapeamento do grafo.

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 Sadik 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.

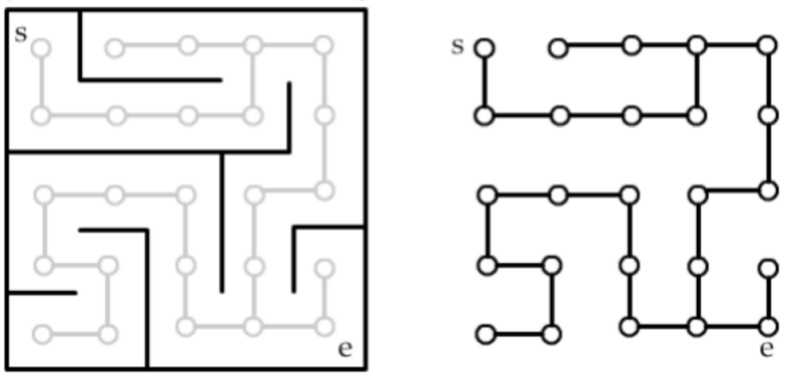

Figura 1. Exemplo de conversão de um labirinto em grafo. Fonte: https://www.cs.umd.edu/class/spring2019/cmsc132-020X-040X/Project8/proj8.html.

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 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.

4.3. Algoritmos de busca não informada

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 (WILSON, 1996, p. 56). 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. Percebe-se então que 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.

Um outro algoritmo bastante utilizado é o *Breadth-First Search* (BFS - em português: busca em largura). Sua busca começa por um vértice inicial e, tendo visitado ele, passa a visitar todos os seus vizinhos para em seguida visitar todos os vizinhos dos vizinhos, e assim por diante (USP). 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 aplicação "pura" dos métodos DFS e BFS a diferença entre eles está na ordem que os vértices são visitados. Isso deriva do tipo de estrutura de dados usado por cada um deles: o DFS utiliza uma pilha, enquanto o BFS utiliza uma fila (USP). Segundo Wilson (2008), não há uma regra definida para qual método deve ser usado para um problema específico, pois cada apresenta suas vantagens - como por exemplo, uma busca em largura é utilizada no algoritmo do caminho mais curto enquanto uma busca em profundidade é usada para encontrar fluxos de rede.

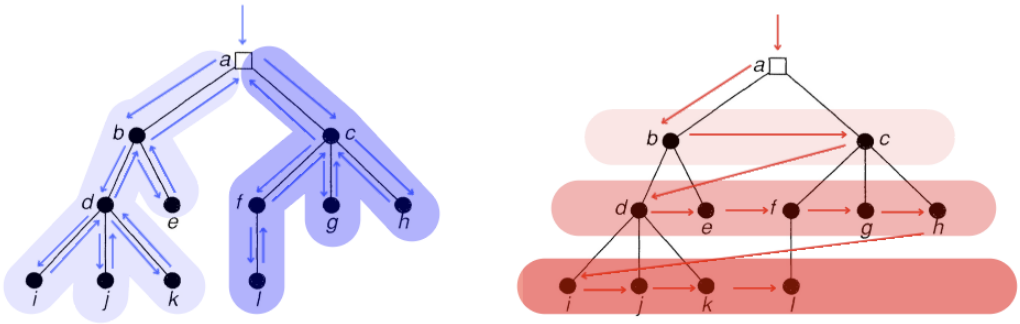

Figura 2. Diferença na ordem de exploração dos algoritmos DFS (esquerda) - ordem ABDIJKECFLGH - e BFS (direita) - ordem ABCDEFGHIJKL. Fonte: Wilson , 2008, p. 56, modificado pelos autores.

É importante também levar em consideração que a exigência de memória para codificação é maior no DFS do que no BFS e o fato do BFS ser mais apropriado para labirintos grandes (Sadik et al., 2010, p. 56).

5. Desenvolvimento

Sabendo 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 foi utilizado inicialmente com a função de mapear o grafo.

Uma vez que o grafo está mapeado, ao buscar o menor caminho possível para achar a saída em um grafo não ponderado, o algoritmo BFS foi escolhido devido à sua eficiência em termos de tempo e garantia de encontrar o menor caminho em um grafo (Sadik et al., 2010, p. 54).

Durante o desenvolvimento desse projeto, buscou-se estratégias para diminuir o tempo de execução do algoritmo. Uma abordagem consistiu em simultaneamente mapear e buscar o caminho, interrompendo a busca assim que o vértice de saída fosse identificado, eliminando a necessidade de percorrer a totalidade do grafo. Entretanto, um dos desafios encontrados nessa estratégia foi a maneira pela qual a API disponibiliza e recebe os dados relativos aos movimentos no grafo.

Figura 3. Formato da requisição e da resposta do *Endpoint* "/movimentar" da API. Fonte: (Graph Theory Maze API, 2023)

Em determinado ponto, o algoritmo BFS encontra uma limitação, pois ao percorrer a largura do grafo, ele pode ficar impossibilitado de alcançar determinadas áreas. Isso ocorre porque, ao seguir uma ramificação do grafo, o algoritmo pode não conseguir transitar para outra devido a restrições na API.

Percebeu-se então que, ao contrário de labirintos predefinidos com estruturas conhecidas, os labirintos gerados pela API não podem ser representados eficientemente como estruturas de matriz. Logo, o labirinto não pode ser mapeado como uma matriz bidimensional de forma direta. Embora todas as informações sobre as arestas do grafo possam ser obtidas a partir de sua matriz de adjacência, algumas informações sobre os vértices podem não ser representadas, como a localização dos vértices no desenho do grafo.

Essa é uma das razões também pela qual uma adaptação do algoritmo A* não foi implementada com sucesso nesse trabalho, pois o cálculo da heurística comumente se baseia em coordenadas cartesianas, o que não pode ser feito com os vértices do grafo fornecido de maneira satisfatória.

Assim, com a aplicação do DFS e, em seguida, o BFS, uma série de testes foram conduzidos para atestar se o grafo inteiro era mapeado e o menor caminho encontrado a partir dos labirintos fornecidos pela API consumida via *web*.

Tabela 1. Avaliação de desempenho do algoritmo desenvolvido

Id 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	3	1914 ms	Sim	Sim
Maze sample 2	10	14	8	9	3	1746 ms	Sim	Sim
Medium maze	40	44	1	22	32	5328 ms	Sim	Sim
Large maze	491	968	161	307	21	59580 ms	Sim	Sim
Very large maze	4675	11511	3421	3567	521	584487 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.

Quanto a eficiência, indica-se aprofundar melhor alternativas para diminuir o tempo de execução do algoritmo com estratégias que eliminem a necessidade de mapeamento por completo do grafo.

É importante destacar que, embora o BFS em sua aplicação pura 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 eficácia 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.

SADIK, M. J.; DHALI, M. A.; FARID, H. M. A. B.; RASHID, T. U.; SYEED, A. 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.

Graph Theory Maze API. Disponível em: https://github.com/rambim/graph_theory_maze. Acesso em: 20 nov. 2023.

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.

USP. Algoritmos para Grafos - Aulas: Busca em largura. Disponível em: https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/bfs.html. Acesso em: 20 nov. 2023.

WILSON, R. J. Introduction to Graph Theory. 4. ed. Essex: Longman, 1996.