AEDS III - TP01

Caio D.Alves[1,2]

[1]Instituto de Ciências Exatas e Aplicadas – Universidade Federal de Ouro Preto (UFOP)
CEP 35931-008 – João Monlevade – MG – Brasil

[2]Departamento de Computação e Sistemas (DECSI)
Universidade Federal de Ouro Preto (UFOP) – João Monlevade, MG – Brasil

{caio}@caio.damasceno@aluno.ufop.edu.br

Abstract. *This practical work aims to develop skills in graph algorithms, focusing on search algorithms and the application of this knowledge to solve real problems. The problem addressed is the resolution of mazes represented by matrices, using Python to find paths between start and end points*

Resumo. *Este trabalho prático tem como objetivo desenvolver habilidades em algoritmos de grafos, focando em algoritmos de busca e na aplicação desses conhecimentos para resolver problemas reais. O problema abordado é a resolução de labirintos representados por matrizes, utilizando Python para encontrar caminhos entre pontos de início e fim.*

1. Introdução

A representação de labirintos é feita através de uma matriz que utiliza caracteres para denotar paredes, passagens, ponto inicial e ponto final. O objetivo deste trabalho é desenvolver um programa capaz de processar essas matrizes, encontrar um caminho entre o ponto inicial e o final, e medir o tempo de execução dos algoritmos utilizados. A avaliação do trabalho considera a precisão e eficiência da implementação.

Para o desenvolvimento deste projeto, a apostila [UFOP 2024] da UFOP foi fundamental, fornecendo uma base sólida em Python para implementar algoritmos de grafos e resolver o problema proposto.

Veja a implementação completa em: Repositório AEDSIII no GitHub

2. Resultados dos Algoritmos BFS e DFS

Esta seção apresenta os tempos de execução dos algoritmos de Busca em Largura (BFS) e Busca em Profundidade (DFS) aplicados a diversos arquivos de labirinto [Veanged 2024]. Os tempos de execução são medidos em segundos.

2.1. Tempos de Execução por Arquivo

- `maze10.txt`
  - **BFS:** 0.00103 s
  - **DFS:** 0.00021 s
  - **Diferença (BFS - DFS):** 0.00082 s
- `maze20.txt`
  - **BFS:** 0.00207 s

    - **DFS:** 0.00085 s
    - **Diferença (BFS - DFS):** 0.00122 s
- `maze30.txt`
    - **BFS:** 0.00345 s
    - **DFS:** 0.00110 s
    - **Diferença (BFS - DFS):** 0.00235 s
- `maze40.txt`
    - **BFS:** 0.01226 s
    - **DFS:** 0.00205 s
    - **Diferença (BFS - DFS):** 0.01021 s
- `maze50.txt`
    - **BFS:** 0.02133 s
    - **DFS:** 0.00495 s
    - **Diferença (BFS - DFS):** 0.01638 s
- `toy.txt`
    - **BFS:** 0.00005 s
    - **DFS:** 0.00003 s
    - **Diferença (BFS - DFS):** 0.00002 s

2.2. Análise das Diferenças de Tempo

A análise dos tempos de execução indica que o algoritmo de Busca em Profundidade (DFS) apresentou um desempenho superior ao algoritmo de Busca em Largura (BFS) em todos os casos. O tempo de execução de DFS foi consistentemente menor que o de BFS. A diferença de tempo entre BFS e DFS aumenta com o tamanho do labirinto, mostrando que DFS é mais eficiente para labirintos maiores.

2.2.1. Comparação por Tamanho do Labirinto

Para labirintos menores, como `maze10.txt` e `maze20.txt`, a diferença entre os tempos de execução de BFS e DFS é pequena. Entretanto, à medida que o tamanho do labirinto aumenta, a diferença se torna mais pronunciada. Por exemplo, para `maze50.txt`, a diferença é de 0.01638 s, um valor significativo comparado aos labirintos menores.

2.2.2. Conclusão sobre o Desempenho

A Busca em Profundidade (DFS) demonstrou ser a escolha mais eficiente em termos de tempo para labirintos grandes, provavelmente devido à sua estratégia de explorar um caminho até o fim antes de retroceder. Essa abordagem é vantajosa em cenários onde a solução é encontrada rapidamente em um caminho profundo.

Em contraste, a Busca em Largura (BFS), que explora todos os caminhos de um nível antes de passar para o próximo, mostrou-se menos eficiente para labirintos maiores devido ao maior número de caminhos a serem explorados. Embora BFS possa ser mais adequado para problemas que requerem a menor distância entre o início e o fim, o desempenho superior de DFS em nossos testes sugere que DFS é preferível para labirintos maiores ou problemas similares.

3. Conclusão

Esta seção sintetiza os resultados obtidos a partir da execução dos algoritmos BFS e DFS nos labirintos testados, discutindo suas implicações e sugerindo direções para futuras pesquisas.

3.1. Resumo dos Resultados

Os testes mostraram que o algoritmo de Busca em Profundidade (DFS) teve um desempenho superior ao algoritmo de Busca em Largura (BFS) em termos de tempo de execução. Em todos os casos analisados, DFS foi mais eficiente, com tempos de execução menores em comparação a BFS. A diferença de tempo aumentou conforme o tamanho do labirinto, indicando uma vantagem crescente para DFS em problemas maiores.

3.2. Implicações dos Resultados

A eficiência temporal superior do DFS pode ser atribuída ao seu método de exploração profunda, que é eficaz para labirintos grandes onde uma solução pode ser encontrada mais rapidamente ao seguir um caminho até o fim antes de considerar outras opções. Em contraste, o BFS, que explora níveis de caminhos, mostrou-se menos eficiente para labirintos grandes devido ao maior número de caminhos a serem explorados.

Esses resultados sugerem que, para problemas que envolvem labirintos ou cenários semelhantes, onde o tempo de execução é crucial, o algoritmo DFS pode ser mais apropriado. No entanto, é essencial considerar as características específicas do problema ao escolher o algoritmo de busca.

3.3. Direções para Futuras Pesquisas

Futuras pesquisas podem explorar os seguintes aspectos:

- **Análise de Complexidade Espacial:** Investigar como a complexidade espacial dos algoritmos DFS e BFS varia com diferentes tipos de labirintos e tamanhos. A eficiência temporal de DFS pode vir com um custo de uso de memória.
- **Variedade de Labirintos:** Testar os algoritmos em uma gama mais ampla de tipos de labirintos, incluindo labirintos com múltiplos caminhos e características variáveis.
- **Otimização dos Algoritmos:** Explorar técnicas de otimização e heurísticas que possam melhorar o desempenho de BFS em problemas que requerem soluções rápidas.
- **Análise Comparativa com Outros Algoritmos:** Comparar BFS e DFS com outros algoritmos de busca, como A* ou algoritmos de busca bidirecional, para uma visão mais completa das opções disponíveis para resolver problemas de labirinto e similares.

Em resumo, enquanto DFS mostrou-se mais eficiente para os labirintos testados, a escolha do algoritmo ideal pode depender das características específicas do problema. A continuação da pesquisa e experimentação com diferentes cenários e algoritmos ajudará a aprimorar nossa compreensão e otimização das técnicas de busca.

Referências

[UFOP 2024] UFOP (2024). Introdução ao python. Acesso em 22 de julho de 2024.

[Veanged 2024] Veanged, M. (2024). Algoritmos em python: Uma análise abrangente e aplicações práticas. *Dev.to*. Acessado em 22 de Julho de 2024.