Análise Comparativa de Algoritmos de Busca na Resolução de Labirintos

Lucas Rodrigues Zabot

19 de abril de 2025

1 Introdução

A resolução de labirintos é um problema clássico na área de Inteligência Artificial e Ciência da Computação, sendo amplamente utilizada para demonstrar a eficiência de algoritmos de busca. Neste trabalho, comparamos quatro algoritmos: Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e A*. O objetivo é analisar o desempenho de cada um quanto ao tempo de execução, número de nós visitados e custo do caminho até a solução.

2 Algoritmos Utilizados

Neste trabalho, foram utilizados quatro algoritmos clássicos de busca aplicados ao problema de encontrar um caminho entre dois pontos dentro de um labirinto. Cada algoritmo possui características próprias, vantagens, desvantagens e aplicações ideais. A seguir, detalhamos o funcionamento de cada um.

2.1 Busca em Largura (BFS)

A Busca em Largura é um algoritmo de exploração que segue a estratégia de visitar primeiro todos os nós vizinhos do ponto inicial, antes de avançar para os vizinhos desses vizinhos, e assim por diante. Em outras palavras, a expansão é feita por *camadas*, garantindo que o caminho encontrado, caso exista, será sempre o mais curto possível em termos de número de passos.

Este algoritmo utiliza uma estrutura de dados do tipo **fila (FIFO)** para gerenciar os nós a serem visitados. Isso significa que os nós são explorados na ordem em que foram descobertos. Por essa razão, a BFS é considerada um algoritmo **completo**, sempre encontra uma solução, se houver, e **otimizado** para grafos não ponderados.

Contudo, sua principal limitação está no alto consumo de memória, já que a fila pode crescer exponencialmente em casos de labirintos grandes ou densamente conectados. Além disso, embora encontre o caminho mais curto, pode demorar para fazê-lo se a solução estiver distante da origem.

2.2 Busca em Profundidade (DFS)

Diferente da BFS, a Busca em Profundidade adota uma abordagem de mergulho em um único caminho até o fim, retornando apenas quando encontra um beco sem saída. Este processo de retroceder e tentar novos caminhos é conhecido como *backtracking*. A DFS utiliza uma **pilha (LIFO)** como estrutura de controle, o que a torna muito eficiente em termos de uso de memória, especialmente em labirintos que possuem muitas ramificações com soluções próximas do início.

No entanto, a DFS não garante que encontrará o caminho mais curto. De fato, ela pode encontrar uma solução subótima ou mesmo se perder em ciclos infinitos se não houver controle sobre os nós já visitados. Ainda assim, é útil para resolver labirintos muito profundos ou quando se deseja encontrar *alguma* solução rapidamente, sem se importar com a qualidade.

2.3 Busca Gulosa

A Busca Gulosa introduz o conceito de **heurística**, ou seja, uma função que estima o quão perto um determinado ponto está do destino. No contexto deste trabalho, foi usada a **distância de Manhattan**, que calcula a soma das distâncias absolutas entre as coordenadas atuais e o objetivo, desconsiderando obstáculos.

A estratégia da busca gulosa é sempre expandir o nó que parecer mais próximo do destino, segundo a heurística. Isso a torna, em muitos casos, extremamente rápida, pois evita explorar caminhos que aparentam estar mais distantes.

Entretanto, essa mesma característica pode ser uma fraqueza. Como ela ignora o custo real já acumulado no caminho, pode seguir trajetórias enganosas que parecem promissoras pela heurística, mas acabam sendo mais longas ou até mesmo sem saída. Por isso, embora eficiente em tempo, não é garantido que encontre o caminho mais curto — ela é rápida, mas não confiável.

2.4 Busca A*

A Busca A* é amplamente considerada como um dos algoritmos de busca mais eficientes e completos em inteligência artificial e ciência da computação. Ela combina o melhor da busca gulosa com o controle da BFS. Utiliza a fórmula:

$$f(n) = g(n) + h(n)$$

onde:

- $g(n)$ representa o custo real para chegar ao nó n a partir da origem;
- $h(n)$ é a heurística estimando o custo de n até o objetivo;
- $f(n)$ é a estimativa total do custo de passar pelo nó n até o destino.

3 Implementação e Configuração dos Testes

A implementação foi realizada em Python 3. Cada labirinto foi carregado a partir de arquivos texto contendo representações visuais com obstáculos e caminhos livres. O script processa a estrutura do labirinto, identifica o ponto de partida (S) e o objetivo (E), e executa os quatro algoritmos.

Os testes foram executados em um ambiente com as seguintes especificações:

- Sistema Operacional: Windows 10
- Processador: Intel Core i7
- Memória RAM: 16 GB
- Python: 3.11+

4 Resultados Comparativos

A Tabela 1 apresenta uma comparação entre os algoritmos para cada instância de labirinto testada, considerando o tempo de execução, o número de nós visitados e o custo do caminho.

Tabela 1: Comparativo dos algoritmos

Algoritmo	Tempo (ms)	Nós Visitados	Custo do Caminho	Memória (MB) / Maze
BFS	0.59	146	28	0.0158 - Maze 1
DFS	0.52	109	108	0.0148 - Maze 1
Gulosa	0.26	29	28	0.0055 - Maze 1
A*	0.6	145	28	0.0111 - Maze 1
BFS	0.64	219	38	0.0225 - Maze 2
DFS	0.57	159	158	0.0218 - Maze 2
Gulosa	0.38	39	38	0.0055 - Maze 2
A*	0.72	175	38	0.0222 - Maze 2
BFS	1.84	1125	78	0.0855 - Maze 3
DFS	1.29	779	778	0.0861 - Maze 3
Gulosa	0.42	79	78	0.0148 - Maze 3
A*	.32	1124	78	0.0883 - Maze 3
BFS	2.15	1279	88	0.1929 - Maze 4
DFS	1.54	929	928	0.1944 - Maze 4
Gulosa	0.56	89	88	0.0148 - Maze 4
A*	2.79	1018	88	0.1141 - Maze 4
BFS	2.93	2072	108	0.2320 - Maze 5
DFS	2.82	1369	1368	0.2343 - Maze 5
Gulosa	0.6	109	108	0.0148 - Maze 5
A*	3.94	1305	108	0.1163 - Maze 5
BFS	3.81	2495	124	0.2320 - Maze 6
DFS	3.07	1673	1672	0.2368 - Maze 6
Gulosa	0.86	125	124	0.0218 - Maze 6
A*	6.99	2494	124	0.2424 - Maze 6

5 Discussão

A tabela apresentada na Seção anterior permite uma análise abrangente sobre o comportamento e eficiência dos quatro algoritmos de busca aplicados em seis labirintos distintos, variando em complexidade e tamanho. Com base nesses dados, é possível observar padrões relevantes que nos ajudam a compreender as vantagens e limitações de cada estratégia.

5.1 Eficiência em Tempo de Execução

De forma geral, a **Busca Gulosa** se destacou como o algoritmo mais rápido em quase todas as instâncias. Isso é consequência de sua natureza heurística, que permite que ela explore diretamente os caminhos aparentemente mais promissores sem considerar muito o custo acumulado. Por exemplo, no Maze 3, ela foi capaz de resolver o problema em apenas 0.42 ms, contra 1.84 ms da BFS e 1.29 ms da DFS.

Contudo, essa rapidez tem seu preço. A Busca Gulosa nem sempre encontra o caminho mais curto — como no Maze 2, em que obteve um custo de caminho 38, superior aos 28 encontrados pelos demais algoritmos.

A **Busca A***, por outro lado, tende a ser a mais demorada, principalmente em labirintos maiores e mais complexos, como nos Mazes 5 e 6, em que seu tempo chegou a 3.94 ms e 6.99 ms, respectivamente. Apesar disso, ela combina o custo acumulado com a heurística, resultando consistentemente no menor

custo de caminho em todas as instâncias, o que reforça sua eficiência no critério de qualidade da solução.

5.2 Uso de Memória

Além do tempo de execução, número de nós visitados e custo do caminho, foi também analisado o uso de memória (em MB) por cada algoritmo. Essa métrica é essencial para avaliar a escalabilidade e viabilidade dos algoritmos em ambientes com recursos limitados.

A **BFS** apresentou o maior consumo de memória na maioria dos labirintos, reflexo da sua estratégia de manter uma fila contendo todos os nós a serem explorados. Isso a torna pouco escalável em ambientes com labirintos amplos ou muitos caminhos possíveis.

A **DFS**, por outro lado, demonstrou um consumo mais econômico, pois utiliza uma pilha com profundidade limitada ao caminho atual. Entretanto, por sua natureza recursiva, pode eventualmente causar estouro de pilha em labirintos muito profundos.

Os algoritmos baseados em heurísticas, **Gulosa** e **A***, exibiram consumo de memória intermediário. Ambos fazem uso de estruturas auxiliares como filas de prioridade e mapas de custo, o que impacta o uso de memória. O **A***, apesar de ter custo um pouco maior que a Gulosa nesse aspecto, apresentou caminhos com custo ótimo, justificando seu uso em situações onde o trade-off entre precisão e desempenho é necessário.

Em resumo, o uso de memória mostrou-se uma métrica valiosa para decisões em ambientes com restrições de hardware, como aplicações em dispositivos móveis, sistemas embarcados ou jogos em tempo real.

5.3 Quantidade de Nós Visitados

A quantidade de nós visitados é um bom indicador do quanto o algoritmo precisou explorar o labirinto. A **DFS**, embora tenha apresentado boa performance de tempo, frequentemente visitou menos nós do que a BFS, porém encontrou caminhos mais longos (como em Maze 1, com custo 108 contra 28 da BFS), o que evidencia sua falta de preocupação com caminhos ótimos.

A **BFS**, por sua vez, sempre encontrou o caminho mais curto possível (em termos de número de passos), mas à custa de visitar muitos nós. Em labirintos maiores (Maze 5, por exemplo), ela visitou mais de 2000 nós. Isso reflete sua natureza exaustiva e mostra por que pode se tornar impraticável em ambientes muito extensos.

5.4 Custo do Caminho

Em termos de qualidade da solução (isto é, o menor custo de caminho), a **Busca A*** foi a mais consistente, igualando ou superando todos os outros algoritmos. A capacidade da A* de balancear a distância já percorrida com uma estimativa heurística do que ainda falta é o que permite essa precisão. Mesmo nos labirintos mais difíceis, como o Maze 6, ela manteve o menor custo de caminho (124), o mesmo das demais estratégias, mas com a vantagem de tomar decisões mais informadas.

Já a **Busca Gulosa**, embora rápida, foi a que apresentou maior variação no custo do caminho, evidenciando sua fragilidade frente a labirintos que exigem um planejamento mais estratégico.

5.5 Escalabilidade

Nos testes com labirintos maiores (Maze 5 e 6), tanto a **BFS** quanto a **DFS** sofreram um aumento considerável no tempo e número de nós visitados. Isso sugere que, embora simples e eficientes em labirintos pequenos, essas abordagens não escalam bem com o aumento da complexidade. Em contrapartida, **A*** e **Gulosa** apresentaram um crescimento mais previsível, evidenciando sua maior adequação a ambientes reais ou aplicações mais complexas.

5.6 Resumo Comparativo

- **BFS**: ótima para encontrar o caminho mais curto, mas com alto custo computacional em grandes labirintos.
- **DFS**: rápida em encontrar qualquer solução, porém não confiável quanto à sua qualidade.
- **Gulosa**: extremamente rápida, mas pode encontrar caminhos ruins.
- **A***: mais completa e eficaz, equilibrando qualidade e desempenho, ideal para aplicações críticas.

6 Conclusão e Trabalhos Futuros

Este estudo demonstrou a aplicabilidade de diferentes algoritmos de busca na resolução de labirintos e as diferenças de desempenho entre eles. Como trabalho futuro, propõe-se:

- Explorar outras heurísticas para A* e Gulosa.
- Aplicar os algoritmos em labirintos tridimensionais.
- Paralelizar os algoritmos para melhor desempenho.
- Visualizar graficamente os caminhos percorridos.

Código-fonte disponível em: `https://github.com/LucasRZabot/-ed02-codigo-`