Implementação dos Algoritmos Busca em Largura(BFS) e A* para Resolver o Problema do Quebra-Cabeça dos 8 Números

João Pedro Ospedal dos Santos

[1]Universidade Tuiuti do Paraná
Curitiba – PR

{joao.santos16}@utp.edu.br

Resumo. *Este trabalho apresenta a implementação dos algoritmos Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e Busca A* para a resolução do problema do quebra-cabeça dos 8 números. A proposta busca comparar a eficiência desses algoritmos com base em dois critérios principais: tempo de execução e uso de memória. O objetivo é avaliar a aplicabilidade prática de cada abordagem e identificar suas vantagens e limitações específicas. A metodologia adotada envolveu o desenvolvimento das soluções em Python, utilizando a IDE Visual Studio Code. Todos os algoritmos foram testados sob as mesmas condições e instâncias de entrada, assegurando a consistência dos resultados. Embora tenha sido possível observar diferenças relevantes entre os métodos, não foi identificada uma única solução claramente superior em todos os aspectos analisados.*

1. Introdução

O problema do quebra-cabeça dos 8 números, também conhecido como 8-puzzle, é amplamente estudado no campo da inteligência artificial por sua capacidade de representar desafios de busca em espaços de estados. Ele consiste em um tabuleiro 3x3 com oito peças numeradas de 1 a 8 e um espaço vazio, permitindo a movimentação das peças até atingir uma configuração final específica. Apesar de sua simplicidade aparente, o problema apresenta uma complexidade significativa quando se considera a vasta quantidade de estados possíveis.

Neste trabalho, implementam-se quatro algoritmos clássicos de busca para a resolução do problema: Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e Busca A*. A escolha desses algoritmos visa explorar abordagens tanto informadas quanto não informadas, permitindo uma comparação prática de seu desempenho em termos de tempo de execução e consumo de memória.

A implementação foi desenvolvida em Python, utilizando a IDE Visual Studio Code, com foco em clareza, padronização e reprodutibilidade dos testes. Cada algoritmo foi submetido às mesmas instâncias do problema, o que garantiu a integridade da comparação entre os resultados obtidos.

2. O Problema do Quebra-Cabeça dos 8 Números

O Quebra-cabeça dos 8 números, ou também conhecido como Quebra-cabeça das 8 fichas, consiste num jogo de tabuleiro contendo 9 espaços, onde 8 são preenchidos por peças enumeradas de 1 a 8, sem repetição e não seguindo uma ordem exata.

O objetivo do jogo é ordenar todas as peças numericamente, deslizando-as para os espaços disponíveis em cada estado do tabuleiro, até ficarem ordenadas de 1 a 8 com o espaço vazio ocupando a última posição no canto inferior direito do tabuleiro [Sandra 2025].

É possível estabelecer condições básicas para a resolução do quebra-cabeça:

1. Estados: A descrição de um estado especifica a localização de cada uma das oito fichas em um dos nove lugares disponíveis. Por motivos de eficiência conbem incluir a localização do espaço vazio.
2. Operadores: O espaço vazio pode moverse à esquerda, à direita, acima ou abaixo.

3. Descrição dos Algoritmos

A Busca em Largura (BFS) explora grafos nível por nível usando uma fila, partindo do nó inicial e visitando vizinhos não explorados até encontrar o objetivo. É completa em grafos finitos, ótima em grafos não ponderados (caminho mais curto), com complexidade de tempo e espaço $O(V + E)$. Ideal para menores distâncias, mas consome muita memória em grafos grandes [Cormen et al. 2009, Russell and Norvig 2020]. Já a Busca em Profundidade (DFS) segue um ramo até o fim antes de retroceder, usando uma pilha. Também tem complexidade $O(V + E)$, mas é mais eficiente em memória ($O(V)$), sendo útil para exploração profunda. Não garante completude em grafos infinitos nem optimalidade, adequada para verificar conectividade [Cormen et al. 2009, Russell and Norvig 2020].

O Algoritmo A*, uma busca informada, usa $f(n) = g(n)+h(n)$ ($g(n)$: custo até o nó; $h(n)$: heurística) e uma fila de prioridade para encontrar o caminho ótimo. Completo e ótimo com heurísticas admissíveis, tem complexidade exponencial $O(b^d)$ no pior caso. É eficiente em planejamento de rotas, mas intensivo em memória [Russell and Norvig 2020, Hart et al. 1968]. Por fim, a Busca Gulosa foca apenas na heurística $h(n)$, ordenando nós por uma fila de prioridade. Rápida, com complexidade potencial $O(b^d)$, não garante completude nem optimalidade, sendo útil quando a heurística é confiável, mas suscetível a resultados inconsistentes [Russell and Norvig 2020, Hart et al. 1968].

Esses algoritmos atendem a diferentes necessidades: BFS e A* priorizam optimalidade, DFS economiza memória, e a Busca Gulosa oferece rapidez. A escolha depende do problema e dos recursos disponíveis.

4. Metodologia

Os algoritmos Busca em Largura (BFS), Busca em Profundidade (DFS), Algoritmo A* e Busca Gulosa foram implementados no Visual Studio Code utilizando Python para resolver o quebra-cabeça das 8 peças, um problema clássico de busca em espaço de estados. Para garantir condições equitativas, os algoritmos foram testados na mesma instância inicial do quebra-cabeça, com profundidades de solução em até 40 movimentos, todas buscando o estado final padrão (peças de 1 a 8, com o espaço vazio na última posição).

A análise comparativa focou na quantidade de movimentos realizada, no tempo de execução e no uso de memória. O tempo de execução, medido em segundos, foi registrado com a biblioteca time, capturando o intervalo desde o início da busca até a solução. O uso de memória, em kbytes (KB), foi monitorado com a biblioteca tracemalloc, registrando o pico de consumo durante cada execução. Os testes foram realizados em um computador

com processador Intel Core i7 de 1.7 GHz, 16 GB de RAM e sistema operacional Windows 11. Os resultados foram organizados em uma tabela para avaliar o desempenho e a escalabilidade de cada algoritmo no problema proposto.

5. Análise dos Resultados Experimentais

Tabela 1. Resultados dos algoritmos no quebra-cabeça das 8 peças.

Algoritmo	Movimentos	Tempo (s)	Uso de Memória (KB)
BFS	13	0.0810	1047.08
DFS	39	0.0573	1341,89
A*	13	0.0025	29,29
Busca Gulosa	37	0.0112	198,05

Os resultados mostram uma clara diferença no número de movimentos necessários para resolver o quebra-cabeça. BFS e A* alcançaram soluções ótimas, utilizando apenas 13 movimentos, enquanto DFS e Busca Gulosa requereram 39 e 37 movimentos, respectivamente. Isso destaca a eficiência do BFS e do A* em encontrar caminhos mais curtos, devido à exploração nivelada do BFS e à heurística admissível do A*, que utiliza a distância de Manhattan.

No que diz respeito ao tempo de execução e ao uso de memória, o A* se destaca como o mais eficiente, com um tempo de 0,0025 s e uso de memória de apenas 29,29 KB, superando os outros algoritmos por ampla margem. O BFS, embora ótimo em movimentos, apresentou tempo de 0,0810 s e elevado consumo de memória (1047,08 KB), refletindo sua abordagem de explorar todos os nós de um nível. O DFS foi mais rápido que o BFS e a Busca Gulosa (0,0573 s), mas consumiu 1341,89 KB de memória. A Busca Gulosa, com 0,0112 s e 198,05 KB, mostrou-se mais eficiente em memória que BFS e DFS, mas ainda distante do A* em ambos os critérios.

Um ponto relevante do experimento foi a imposição de um limite de profundidade ao DFS para evitar loops infinitos, uma vez que o algoritmo pode revisitar estados cíclicos no quebra-cabeça. O número de movimentos do DFS (39) aproximou-se desse limite, sugerindo que, em algumas instâncias ou com limites mais restritivos, o algoritmo pode não encontrar a solução. A Busca Gulosa, embora tenha demandado número de movimentos próximo ao DFS (37), não apresentou risco de loops, além de ser mais rápida que o BFS e ter uso de memória significativamente menor, ficando atrás apenas do A*. Esses resultados indicam que o A* é o mais eficiente para o quebra-cabeça das 8 peças, equilibrando optimalidade, tempo e uso de memória, enquanto DFS e Busca Gulosa enfrentam limitações em cenários que exigem caminhos mais curtos.

6. Conclusão

A análise dos algoritmos Busca em Largura (BFS), Busca em Profundidade (DFS), Algoritmo A* e Busca Gulosa no quebra-cabeça das 8 peças revelou diferenças significativas em desempenho. BFS e A* destacaram-se por encontrar soluções ótimas, requerindo apenas 13 movimentos, enquanto DFS e Busca Gulosa utilizaram 39 e 37 movimentos, respectivamente, indicando menor eficiência em encontrar caminhos curtos. Em termos

de tempo de execução e uso de memória, o A* foi superior, com 0,0025 s e 29,29 KB, seguido pela Busca Gulosa (0,0112 s e 198,05 KB), enquanto BFS (0,0810 s e 1047,08 KB) e DFS (0,0573 s e 1341,89 KB) apresentaram maior consumo de recursos. O limite de profundidade imposto ao DFS evitou loops infinitos, mas expôs sua limitação em instâncias complexas. Assim, o A* mostrou-se a melhor escolha para o quebra-cabeça das 8 peças, equilibrando optimalidade, rapidez e baixo uso de memória, enquanto os outros algoritmos podem ser mais adequados para cenários com diferentes restrições ou objetivos.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

Sandra (2025). 8-puzzle. `https://sites.icmc.usp.br/sandra/G5_t2/8_Puzzle.htm`. Acessado em 14 de abril de 2025.