Implementação do Jogo de 8 Peças

Vinícius Miranda de Araújo
Pontificia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática

RESUMO

Este trabalho apresenta a implementação e avaliação de diferentes algoritmos de busca aplicados ao jogo do quebra-cabeça de 8 peças (8-puzzle). Foram testados os algoritmos Breadth-First Search (BFS), Depth-First Search (DFS), Uniform Cost Search (UCS), A* com heurísticas de Manhattan e peças fora do lugar, e Greedy com as mesmas heurísticas. Três cenários com diferentes dificuldades foram utilizados: fácil, médio e difícil. Os testes incluíram medições de tempo de execução, uso de memória e número de passos necessários para alcançar a solução. Os resultados demonstram que os algoritmos baseados em heurísticas, especialmente A* com Manhattan, apresentaram o melhor desempenho em termos de eficiência e escalabilidade, resolvendo até os casos mais difíceis com rapidez e uso moderado de recursos. A implementação está disponibilizada no github: https://github.com/vinimiraa/8puzzle

1. INTRODUÇÃO

O presente trabalho tem como objetivo implementar e comparar diversos algoritmos de busca aplicados ao jogo do 8-puzzle. O problema consiste em mover peças numeradas em um tabuleiro 3x3 até atingir uma configuração objetivo, utilizando o menor número de movimentos possível. A análise considera desempenho em tempo de execução, uso de memória, número de passos para a solução e escalabilidade dos métodos.

2. ALGORITMOS

Foram implementados e comparados diferentes algoritmos de busca para resolver o problema do 8-puzzle. Cada algoritmo adota uma estratégia distinta de exploração do espaço de estados:

- Busca em Largura (BFS - Breadth-First Search): Explora os estados em ordem crescente de profundidade. Garante encontrar a solução com o menor número de movimentos, mas consome muita memória, especialmente em instâncias mais complexas.

- Busca em Profundidade (DFS - Depth-First Search): Explora os estados seguindo o caminho mais profundo antes de retroceder. É mais eficiente em termos de memória do que a BFS, porém não garante encontrar a solução ótima, e pode entrar em ciclos se não houver controle de estados visitados.

- Busca Custo Uniforme (UCS - Uniform Cost

Search): Explora os estados com base no menor custo acumulado, ignorando qualquer heurística. Funciona de forma semelhante ao A* sem heurística, garantindo a solução ótima, mas com custo computacional elevado.

- Busca A* (A-Star): Combina o custo acumulado (g(n)) com a estimativa de custo restante até o objetivo (h(n)), sendo h(n) uma heurística admissível. É eficiente e encontra soluções ótimas.

- Busca Gulosa (Greedy Best-First Search): Utiliza apenas a heurística h(n), ignorando o custo acumulado. Embora seja mais rápida em muitos casos, não garante a solução ótima.

Cada algoritmo foi testado em três níveis de dificuldade de configuração inicial do tabuleiro (Fácil, Médio e Difícil), para análise comparativa de desempenho em termos de tempo, memória e número de passos até a solução.

3. HEURÍSTICAS

Para os algoritmos informados, foram aplicadas duas heurísticas clássicas no contexto do problema do 8-puzzle: Distância de Manhattan e Número de Peças Fora do Lugar (Misplaced Tiles).

- Distância de Manhattan: Essa heurística calcula, para cada peça, a soma das distâncias horizontais e verticais entre sua posição atual e a posição correta no estado objetivo. É considerada admissível e consistente, sendo amplamente utilizada devido à sua capacidade de fornecer uma estimativa mais precisa do custo restante até a solução. Por isso, tende a guiar melhor os algoritmos de busca, como o A*.

- Número de Peças Fora do Lugar: Essa heurística simplesmente conta quantas peças estão fora da posição correta, desconsiderando a distância que precisam percorrer. Embora seja mais simples e computacionalmente leve, geralmente oferece estimativas menos informativas que a distância de Manhattan, podendo levar os algoritmos a explorar mais nós antes de encontrar a solução.

Essas heurísticas foram utilizadas tanto nos algoritmos A* quanto Greedy, e seus desempenhos comparativos foram avaliados nos diferentes níveis de dificuldade do jogo.

4. RESULTADOS EXPERIMENTAIS

Foram testados três casos: fácil, médio e difícil.

4.1. CASO FÁCIL

Algoritmo	Tempo (s)	Memória (KB)	Passos
BFS	0.0010	13.98	2
DFS	0.0000	3.03	2
A_STAR (MANHATTAN)	0.0010	4.64	2
A_STAR (MISPLACED)	0.0000	4.59	2
GREEDY	0.0000	3.57	2

Algoritmo	Tempo (s)	Memória (KB)	Passos
(MANHATTAN)			
GREEDY (MISPLACED)	0.0000	3.57	2
UCS	0.0026	11.03	2

4.2. CASO MÉDIO

Algoritmo	Tempo (s)	Memória (KB)	Passos
BFS	0.0010	10.36	2
DFS	13.6426	56851.62	48
A_STAR (MANHATTAN)	0.0010	3.27	2
A_STAR (MISPLACED)	0.0010	3.15	2
GREEDY (MANHATTAN)	0.0010	3.16	2
GREEDY (MISPLACED)	0.0010	3.16	2
UCS	0.0030	6.80	2

4.3. CASO DIFÍCIL

Algoritmo	Tempo (s)	Memória (KB)	Passos
BFS	9.0831	42708.54	20
DFS	7.0120	38843.93	92
A_STAR (MANHATTAN)	0.0233	122.18	20
A_STAR (MISPLACED)	0.4846	2565.66	20
GREEDY	0.0223	112.75	48

Algoritmo	Tempo (s)	Memória (KB)	Passos
(MANHATTAN)			
GREEDY (MISPLACED)	0.2221	1063.88	116
UCS	9.9604	46302.96	20

5. DISCUSSÃO E COMPARAÇÃO

Observa-se que os algoritmos A* com heurística de Manhattan são os mais eficientes em tempo e uso de memória para todos os casos testados. A busca em profundidade (DFS) apresentou desempenho inconsistente, especialmente no caso Medium, com consumo elevado de memória e tempo.

Greedy com heurísticas não garantem o menor caminho, como evidenciado nos passos das soluções. A UCS e BFS obtiveram soluções ótimas, mas com alto custo de memória em casos difíceis. A heurística de Manhattan foi mais eficaz que a de peças deslocadas em A* e Greedy.

6. CONCLUSÃO

Neste trabalho, foi realizada a implementação e comparação de diversos algoritmos de busca aplicados à resolução do jogo das 8 peças. Os métodos testados incluíram buscas não informadas (BFS, DFS, UCS) e informadas (A*, Greedy) com duas heurísticas distintas: número de peças fora do lugar e distância de Manhattan.

Os resultados experimentais demonstraram que o algoritmo A* com a heurística de Manhattan

apresentou o melhor desempenho geral, oferecendo soluções ótimas com o menor tempo de execução e consumo moderado de memória, mesmo para casos difíceis. A busca gulosa, apesar de mais rápida em alguns casos, frequentemente gerou soluções subótimas e com maior número de passos. A DFS, embora eficiente em problemas simples, mostrou-se ineficaz e com alto consumo de memória em problemas mais complexos, enquanto o UCS e o BFS se destacaram pela precisão, porém com alto custo computacional em instâncias difíceis.

Conclui-se, portanto, que a escolha adequada do algoritmo e da heurística é essencial para garantir desempenho e escalabilidade na resolução de problemas de busca, como o 8-puzzle. Como trabalho futuro, recomenda-se explorar variantes como IDA* e otimizações com estruturas de dados mais eficientes.