Breno Pires Santos

Resumo

Este documento descreve o desenvolvimento e a análise de desempenho de diferentes algoritmos de busca aplicados ao tradicional jogo de quebra-cabeça conhecido como 8-puzzle. Os métodos testados incluíram: busca em largura (BFS), busca em profundidade (DFS), busca de custo uniforme (UCS), algoritmo A* com heurísticas de Manhattan e peças fora do lugar, e busca gulosa (Greedy) com as mesmas heurísticas. A avaliação considerou três níveis de dificuldade — fácil, médio e difícil — com medições de tempo de execução, consumo de memória e quantidade de movimentos necessários para atingir a configuração final. Os resultados indicam que os algoritmos baseados em heurísticas, especialmente o A* com distância de Manhattan, foram os mais eficientes e escaláveis, conseguindo resolver até os desafios mais complexos com rapidez e consumo razoável de recursos. O código-fonte está disponível em: https://github.com/brenodft/CC/tree/main/IA/puzzle

Introdução

O objetivo deste estudo foi implementar e comparar diversos algoritmos de busca para a resolução do quebra-cabeça das 8 peças. O jogo consiste em reorganizar peças numeradas dentro de um tabuleiro 3x3 até atingir a configuração correta com o menor número possível de movimentos. A análise realizada levou em conta critérios como tempo de execução, uso de memória, quantidade de passos até a solução e a escalabilidade dos algoritmos.

Algoritmos

Foram utilizados diferentes algoritmos para explorar o espaço de estados do problema, cada um com sua estratégia própria:

- **Busca em Largura (BFS)**: Explora os estados com base na profundidade crescente. É garantido que encontra a solução ótima em termos de número de movimentos, mas possui alto consumo de memória em casos complexos.
- **Busca em Profundidade (DFS)**: Prioriza o caminho mais profundo primeiro. Tem baixo consumo de memória, mas não assegura a solução ótima e pode entrar em ciclos caso não haja controle dos estados visitados.
- **Busca A***: Usa o custo acumulado (g(n)) mais uma estimativa do custo restante (h(n)), com base em heurísticas admissíveis. É eficiente e encontra soluções ótimas.

Todos os algoritmos foram testados com configurações iniciais fáceis, médias e difíceis, avaliando tempo, memória e número de passos para resolução.

Heurísticas

As duas heurísticas utilizadas foram:

- **Distância de Manhattan**: Calcula, para cada peça, a soma das distâncias horizontal e vertical até sua posição correta. É considerada precisa, admissível e consistente, o que a torna eficaz para o algoritmo A*.
- **Peças Fora do Lugar**: Simplesmente contabiliza as peças fora da posição correta. É computacionalmente mais leve, mas menos informativa que a distância de Manhattan.

Resultados

Algoritmo	Tempo(s)	Memória(KB)
BFS	0.0010	14
DFS	0.0000	3
A* Manhattan	0.0010	4.6
A* Misplaced	0.0000	4.5