Conteúdo

[Resolução do 8-Puzzle 2](#_Toc191281436)

[Descrição do Puzzle: 2](#_Toc191281437)

[Descrição dos Algoritmos Implementados: 2](#_Toc191281438)

[A\* com Heurística Manhattan: 2](#_Toc191281439)

[A\* com Heurística Hamming: 2](#_Toc191281440)

[A\* com Manhattan 3](#_Toc191281441)

[A\* com Hamming 3](#_Toc191281442)

[Pesquisa em Largura (BFS) 3](#_Toc191281443)

[Estudo de Custo de Tempo e Memória 3](#_Toc191281444)

[Discussão dos Resultados 4](#_Toc191281445)

[Conclusões Principais 6](#_Toc191281446)

# Resolução do 8-Puzzle

## Descrição do Puzzle:

O 8-Puzzle é um quebra-cabeça deslizante composto por 8 peças numeradas (de 1 a 8) distribuídas em um tabuleiro 3x3, onde uma posição é deixada vazia (representada por 0 ou espaço em branco). O objetivo do jogo é transformar uma configuração inicial numa configuração final (objetivo) através de movimentos válidos, deslizando as peças para ocupar a posição vazia. A natureza do puzzle gera um espaço de estados finito, porém com complexidade combinatória considerável, o que torna a sua resolução um problema clássico de pesquisa em inteligência artificial.

## Descrição dos Algoritmos Implementados:

O código contempla diferentes abordagens para a resolução do puzzle:

Modo Manual: Permite que o utilizador mova as peças utilizando as setas do teclado. Cada movimento é executado interactivamente e o número de movimentos é contabilizado.

Modo Automático: O utilizador pode escolher entre três algoritmos para resolver o puzzle automaticamente:

### A\* com Heurística Manhattan:

Utiliza a soma das distâncias horizontais e verticais de cada peça até à sua posição correta na configuração alvo. Esta heurística é admissível e consistente, fornecendo uma boa orientação e, normalmente, gerando a solução ótima com eficiência.

### A\* com Heurística Hamming:

Calcula o número de peças fora de posição (desconsiderando o espaço vazio). Embora também seja admissível, esta heurística é menos informativa do que a Manhattan, o que pode levar a uma expansão maior de nós durante a pesquisa.

Pesquisa em Largura (BFS): Explora os estados do puzzle de forma nivelada (por camadas), garantindo encontrar a solução com o menor número de movimentos (solução ótima em termos de custo uniforme). No entanto, BFS pode tornar-se impraticável em termos de tempo e memória para estados com maior profundidade, devido à explosão combinatória do espaço de estados.

Principais Características de Cada Algoritmo

### A\* com Manhattan

Optimalidade: Garante encontrar a solução ótima, desde que a heurística seja admissível e consistente (o que é o caso da Manhattan).

Completude: É completo, desde que o espaço de estados seja finito.

Complexidade: Em geral, a complexidade é exponencial no pior caso; contudo, a boa orientação da heurística Manhattan costuma reduzir significativamente o número de nós expandidos na prática.

### A\* com Hamming

Optimalidade: Também encontra a solução ótima, pois a heurística Hamming é admissível.

Completude: Completo para espaços finitos.

Complexidade: Pode ser menos eficiente do que a Manhattan, pois a contagem de peças fora do lugar fornece uma estimativa menos refinada do custo restante, aumentando o número de estados explorados.

### Pesquisa em Largura (BFS)

Optimalidade: Garante encontrar a solução com o menor número de movimentos, pois explora o espaço de estados de forma ordenada por profundidade.

Completude: É completo para espaços de estados finitos.

Complexidade: Apresenta uma complexidade de tempo e espaço exponencial; embora seja simples de implementar e garantir a optimalidade, o uso intensivo de memória torna-o inviável para instâncias mais complexas do puzzle.

## Estudo de Custo de Tempo e Memória

BFS: Explora um grande número de estados, implicando um alto custo em tempo de execução e consumo de memória.

A\* com Hamming: Expande mais nós do que a Manhattan, tornando-se menos eficiente em puzzles mais complexos.

A\* com Manhattan: Garante menor tempo de execução e uso reduzido de memória, pois menos nós são expandidos até à solução.

## Discussão dos Resultados

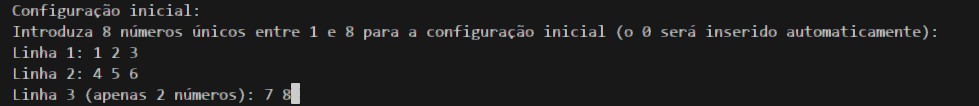
Eficiência e Optimalidade: Todos os algoritmos garantem optimalidade, mas variam em eficiência.

Escalabilidade: A\* demonstra melhor escalabilidade em comparação com BFS.

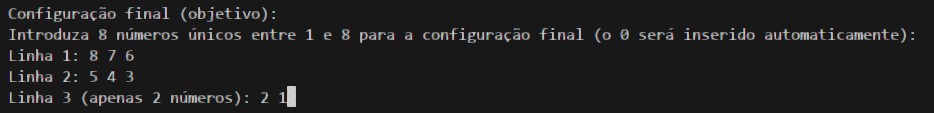
Aplicabilidade: A\* com Manhattan é a melhor escolha na maioria dos cenários.

Combinações Testadas e Respostas A seguir, apresentamos cinco exemplos de combinações testadas e as respostas obtidas:

Caso Inicial



Caso Final

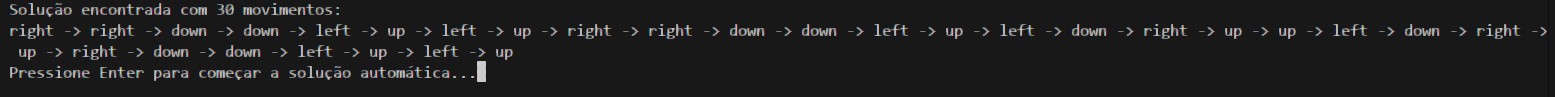


Escolha de qual queremos usar

Uma imagem com texto, captura de ecrã, Tipo de letra, preto

Os conteúdos gerados por IA poderão estar incorretos.

Solução



Solução apos a resolução automática

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

Comparação das 3 opções

Uma imagem com texto, captura de ecrã, preto, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

## Conclusões Principais

Versatilidade do Código: Modos interativos e automáticos permitem experiência prática e análise de desempenho.

Importância da Heurística: A heurística Manhattan reduz significativamente o espaço de pesquisa.

Trade-off entre Simplicidade e Desempenho: BFS é simples mas ineficiente para problemas complexos.

Aplicação em Problemas Reais: A seleção de heurísticas adequadas é crucial para o desempenho dos algoritmos de pesquisa em inteligência artificial.