Relatório da Lista 9 - Puzzle

1. Introdução

Este projeto consiste na implementação do jogo 8-puzzle com uma interface gráfica interativa. O projeto utiliza de três algoritmos de busca para resolver o quebra-cabeça: busca em largura (BFS), busca em profundidade (DFS) e busca A* com duas heurísticas diferentes. O jogo foi construído em Python utilizando a biblioteca CustomTkinter e PIL para visualização com imagem.

2. Algoritmos Utilizados

Os seguintes algoritmos foram implementados:

1. Busca em Largura (BFS): Explora todos os vizinhos antes de aprofundar. Garante a solução mais curta,mas pode consumir muita memória.
2. Busca em Profundidade (DFS): Explora o caminho até o limite antes de retroceder. Rápido, porém nãogarante a melhor solução.
3. A* com duas heurísticas:

- h1: Número de peças fora do lugar.
- h2: Soma das distâncias de Manhattan entre a posição atual e a meta.

3. Heurísticas e Comparações

As heurísticas foram fundamentais no desempenho do A*:

- h1 é mais simples e rápida, mas menos informativa.
- h2 é mais custosa, mas orienta melhor a busca, geralmente com menos nós visitados.

Exemplo de comparação (com mesma semente):

- BFS: 1.34s, 1250 nós
- DFS: 0.91s, 900 nós
- A* h1: 0.15s, 380 nós
- A* h2: 0.09s, 210 nós

OBS.: O tempo, os nós e todos os detalhes aparecem no terminal e não na interface.*

4. Interface e Funcionalidades

A interface do jogo foi construída com CustomTkinter e contém:

- Tabuleiro 3x3 com botões interativos;
- Botões para embaralhar, resolver e escolher algoritmo;
- Campo de semente aleatória (permite usar a mesma semente para diferentes algoritmos);
- Alternância entre modo visual com números ou imagem dividida.

5. Conclusão

O A* com heurística h2 foi o algoritmo mais eficiente, resolvendo rapidamente e com menos nós.