UNIVERSIDADE DE FORTALEZA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
CURSO: CIÊNCIA DA COMPUTAÇÃO

Samuel Duarte Fernandes Lima – 2216976

1. INTRODUÇÃO

Demonstração dos algoritmos aplicados no jogo 8/15 Puzzle: Breadth-first Search (BFS), Depth-first Search (DFS), Iterative Depth-first Search (IDFS), Greedy e Hill Climbing. A matriz objetivo é embaralhada com um número específico de movimentos, neste caso apenas 150, gerando uma configuração inicial de forma aleatória e única, o que significa que, embora a sequência de movimentos seja aleatória, ela nunca gera duas configurações iguais. Isso ocorre porque, mesmo sendo gerados aleatoriamente, os movimentos seguem uma sequência única, garantindo que a configuração final não se repita. Essa abordagem visa facilitar a observação do comportamento dos algoritmos durante o jogo, evitando que possam levar a ciclos infinitos ou soluções inviáveis.

Sendo as matrizes objetivos:

1	2	3
8		4
7	6	5

1	2	3	4
12	13	14	5
11		15	6
10	9	8	7

2. HEURÍSTICAS

As heurísticas escolhidas foram: A primeira, baseada no número de peças que estão na mesma posição da matriz objetivo, avaliando quantas peças já estão corretamente posicionadas; A segunda é a de Manhattan, que calcula a soma das distâncias horizontais e verticais de cada peça fora de lugar até sua posição correta no tabuleiro.

3. COMPARATIVO DE RESULTADOS

BFS (Breadth-first Search)

- **Tamanho máximo:** 26.300
- **Tempo de execução:** 1.69 segundos

No exemplo, ele teve um tamanho máximo de memória de 26.300, consumindo 1.69 segundos, o que indica um desempenho relativamente aceitável para buscas cegas, mas com um custo em termos de consumo de memória, uma vez que mantém em memória todos os nós explorados.

DFS (Deapth-first Search)

- **Tamanho máximo:** 69.300
- **Tempo de execução:** 40.69 segundos

Nos testes realizados, apresentou um desempenho muito inferior em termos de memória e tempo de execução. Em alguns casos, o DFS demorou quase 90 segundos para encontrar uma solução, tornando-se impraticável para problemas mais complexos.

IDFS (Iterative Deepening Depth-first Search)

- **Tamanho máximo:** 24
- **Tempo de execução:** 1.85 segundos

Apresentou um tamanho máximo de memória significativamente menor, apenas 24, comparado aos outros algoritmos. Porém, o tempo de execução foi maior que o BFS, com 1,85 segundos.

Greedy Tile Matching

- **Tamanho máximo:** 46.511
- **Tempo de execução:** 2.24 segundos

O algoritmo Greedy com a heurística de Tile Matching, que conta quantas peças estão no lugar correto, utilizou 46.511 de memória em um tempo de 2,24 segundos. Esse método, por ser guloso, tenta sempre o movimento que parece mais promissor a curto prazo, mas não garante a solução ótima e pode acabar explorando muitos caminhos desnecessários.

Greedy Manhattan

- **Tamanho máximo:** 46.511
- **Tempo de execução:** 3,25 segundos

Apresentou 46.511 de uso máixmo de memória, semelhante ao Greedy Tile Matching. Contudo, o tempo de execução foi maior, 3,25 segundos, indicando que o cálculo da heurística de Manhattan, apesar de ser mais informativo, é mais custoso em termos de tempo. Essa heurística, por ser mais complexa, levou mais tempo, resultando em um tempo final maior, mas sem uma redução no número de máximo de memória.

Hill Climbing

O algoritmo Hill Climbing, que também foi testado, demonstrou que, na maioria dos casos, ele não encontra uma solução para o problema. Isso ocorre porque o Hill Climbing, sendo um algoritmo guloso local, frequentemente fica preso em máximos locais — estados onde nenhum movimento aparente melhora a situação, mesmo que ainda esteja longe do objetivo. Devido a essa limitação, não foi possível mensurar adequadamente o desempenho desse algoritmo, já que ele raramente chegava ao objetivo.

4. CONCLUSÃO

Com base nos testes realizados para o jogo do 8/15 puzzle, o IDFS (Iterative Deepening Depth-First Search) apresentou o melhor desempenho em termos de uso de memória, encontrando a solução com apenas 24 passos, enquanto os outros algoritmos, como o BFS e os métodos gulosos, exigiram um número significativamente maior uso. Embora o tempo de execução do IDFS tenha sido ligeiramente superior ao BFS, seu baixo consumo de memória e a eficiência na busca justificam sua aparente superioridade. Entretanto, é importante destacar que essa conclusão não reflete necessariamente a realidade em outros cenários, O **IDFS** pode sofrer um aumento significativo no tempo de execução à medida que o espaço de busca cresce, e algoritmos como o BFS ou até heurísticas mais elaboradas, como as usadas em algoritmos gulosos, podem ser mais adequados dependendo do tamanho e da natureza do problema. Assim, a escolha do algoritmo ideal depende fortemente das características específicas do problema e dos recursos disponíveis.