**Universidade Federal de Minas Gerais**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

**DCC642 Introdução à IA**
**1º Semestre de 2022**
**Prof. Luiz Chaimowicz**

# Trabalho Prático 1

## 8-Puzzle

**Data de entrega: 01/06**

O objetivo do trabalho consiste em implementar e comparar os diferentes métodos de busca apresentados durante o curso, aplicando-os a um *toy problem*. O quebra-cabeça das oito peças (8-Puzzle) é composto por uma moldura 3x3 contendo um conjunto de peças numeradas de 1 a 8 e um espaço vazio. O propósito do jogo resume-se a posicionar as peças em uma determinada ordem (Figura 1) apenas deslizando-as pela moldura. Sabe-se que o problema de encontrar uma solução com o menor número de passos no caso geral, denominado N-Puzzle, é NP-Difícil [1].

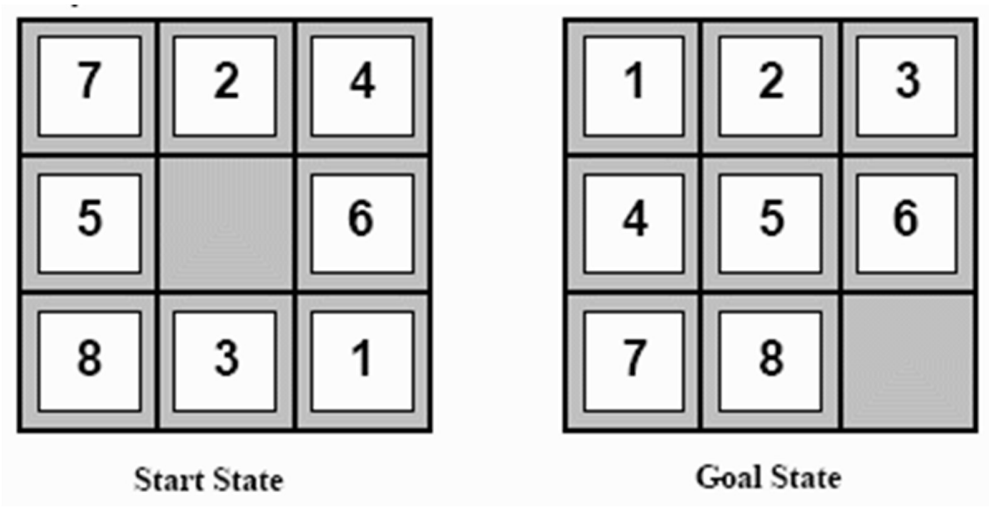


***Figura 1.*** *Estados inicial e final para o 8-Puzzle.*

Vocês devem implementar, em uma das seguintes linguagens (C, C++, Java ou Python), as estruturas de dados, heurísticas e algoritmos necessários para resolver esse quebra-cabeças. Particularmente, os seguintes algoritmos de busca devem ser implementados e comparados:

| | |
|---|---|
| **Busca sem informação** | *Breadth-first search*<br>*Iterative deepening search*<br>*Uniform-cost search* |
| **Busca com informação** | *A* search*<br>*Greedy best-first search* |
| **Busca local** | *Hill Climbing*, permitindo movimentos laterais. |

Com relação aos algoritmos A* e Greedy best-first, vocês devem utilizar **duas heurísticas distintas**.

Já o Hill Climbing vai ter que ser ligeiramente alterado para salvar o caminho encontrado e permitir movimentos "laterais" no espaço de estados até um limite *k*. Determine um valor k adequado.

Um arquivo com conjunto de entradas e o número de passos para a solução está disponível (npuzzle.pdf).

**Formato de Entrada e Saída:**

Para facilitar a correção, o seu programa deverá se chamar TP1 e ler os parâmetros a partir da linha de comando. O primeiro parâmetro deverá ser o algoritmo a ser utilizado (B, I, U, A, G, H), seguido da configuração da entrada (as 3 linhas do 8 puzzle em sequência, usando o número 0 para representar o espaço vazio). E opcionalmente um último parâmetro (PRINT) indicando se os passos intermediários até a solução devem ser impressos.

Por exemplo, para resolver o problema mostrado na Figura 1 usando o A*, a sua entrada deverá ser:

```
% TP1 A 7 2 4 5 0 6 8 3 1
```

A saída deverá ser somente o número de passos para a solução. Se a opção PRINT for utilizada, você deverá imprimir o número de passos e em seguida todos os estados intermediários até a solução. Por exemplo, para a entrada com tamanho de solução 2 do arquivo npuzzle.pdf, deverá ser impresso:

```
% TP1 B 1 2 3 4 0 5 7 8 6 PRINT
2

1 2 3
4   5
7 8 6

1 2 3
4 5
7 8 6

1 2 3
4 5 6
7 8
```

**O que deve ser entregue:**

Faça um .zip ou similar contendo

- Códigos fonte dos algoritmos desenvolvidos
- Um arquivo readme.txt com informações sobre como compilar e executar os seus programas
- Documentação contendo uma **descrição sucinta** das estruturas de dados, heurísticas e algoritmos empregados para modelar e resolver o problema, como também uma **discussão dos resultados obtidos.**

Mais especificamente, esse documento deve incluir:

- Apresentação das estruturas usadas e da modelagem dos componentes da busca (estado, função sucessora, etc);
- Breve descrição das principais diferenças entre os algoritmos;
- Especificação das heurísticas utilizadas. Elas são admissíveis? Por quê?;
- Exemplos de soluções encontradas pelos algoritmos.
- Análise quantitativa comparando os algoritmos com relação ao número de estados expandidos e tempo de execução à medida que o número de passos para a solução aumenta. **Apresente tabelas e/ou gráficos comparativos.**
- Discussão dos resultados obtidos.

**Bom trabalho!**

[1] Daniel Ratner, Manfred K. Warmuth. *Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable*. National Conference on Artificial Intelligence, 1986.