ATIVIDADE DE LÓGICA PARA COMPUTAÇÃO

Prof. Alexandre Arruda

Limite de Participantes: **3** (mais que isso haverá penalização 20% por cada novo membro)

Data limite para entrega: **27/07/2025**

Linguagem: python

Obs.: Plágio e uso de IA's não serão permitidos (penalidade: **reprovação**)

Descrição:

Este trabalho tem como objetivo a implementação de um programa que resolva o problema 8-puzzle.

O 8-puzzle é um quebra-cabeça deslizante que consiste em uma grade 3x3 com 8 peças numeradas (de 1 a 8) e um espaço vazio (representado por 0). O objetivo é reorganizar as peças, movendo-as para o espaço vazio, até que a ordem numérica seja alcançada na grade. As peças só podem ser movidas na horizontal ou vertical para o espaço vazio.

1	2	3
8	0	4
7	6	5

Estado Inicial

0	1	2
3	4	5
6	7	8

Estado Final

Você deverá utilizar o solver python-sat (https://github.com/pysathq/pysat) para criar uma solução que, a partir de um estado inicial, indique como obter o estado final por meio da movimentação das peças.

Dicas:

Crie símbolos proposicionais e mapeie-os 1 a 1 para inteiros positivos.

Um exemplo é: 1_P_2_3_9

onde o número 1 indica que é o primeiro estado (estado inicial), e os números 2 e 3 representam as coordenadas do número 9 no puzzle.

O mapeamento pode ser feito utilizando um dict em python, por exemplo,
1_P_1_1_0 -> 1
1_P_1_1_1 -> 2
…
O mapeamento serve para utilizar o solver proposto.

Crie regras para cada estado

No estado inicial se um número estiver na primeira posição (1,1) certamente nenhum outro número poderá estar na mesma posição. Teremos, portanto,
1_P_1_1_0 -> ¬1_P_1_1_1
1_P_1_1_0 -> ¬1_P_1_1_2
…
1_P_1_1_9 -> ¬1_P_1_1_8

e também devemos ter que pelo menos um número que deve ocupar a posição, o que pode ser feito com a disjunção
(1_P_1_1_0 v … v 1_P_1_1_9)

Crie símbolos para as ações de transição
Exemplo: Uma ação 1_A_D indica que no estado inicial (1) a ação de 0 (vazio) ir para a direita foi realizada. As ações devem também ser mapeadas para inteiros a fim de se obter a sequência de movimentações obtidas pelo solver. **Só uma única ação pode ser feita em um dado estado**.

Crie regras de transição de estado
Se o número 4 estiver na posição (1,3) e o vazio estiver na posição (2,3) uma ação de mover o 0 (vazio) para cima (C) deve fazer o 0 ocupar posição (1,3) e o 4 ocupar a (2,3). Se estivermos no estado inicial isso pode ser feito com a regra

1_P_1_3_4 & 1_P_2_3_0 & 1_A_C -> 2_P_1_3_0 & 2_P_2_3_4

Outra dica.: Se uma movimentação for impossível pode-se repetir o mesmo estado.

Crie uma fórmula que indique se uma solução pode ser encontrada com N movimentações.

Por exemplo, uma fórmula 8_P_1_1_0 & 8_P_1_2_1 … & 8_P_3_3_8 indica que uma solução foi encontrada com 8 passos. Tente achar uma solução ótima varrendo do estado inicial até um número N de passos (ou seja, o solver será utilizado várias vezes).

Importante: O estado inicial deve ser gerado a partir do estado meta, com movimentos aleatórios, caso contrário você pode ter o risco de lidar com um estado inicial sem solução.

O que será avaliado:
- **Apresentação dos integrantes do grupo**
- **Legibilidade do código apresentado**
- **Resultados obtidos**
- **Criatividade (Representação gráfica por exemplo)**