Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

Escola Superior de Tecnologia
Curso de 1º Ciclo em Engenharia Informática
1º Semestre do 2º Ano
Unidade Curricular: Inteligência Artificial
Docente: Ana Paula Neves Ferreira da Silva

O problema do Puzzle de 8 peças - continuação

1. Representação do Puzzle de 8 peças e utilização de algoritmos de procura para a sua resolução

Recordar:

[[0,0], [2, 0], [2, 2], [1, 1], [1, 2], [1, 0], [0, 2], [0, 1], [2,1]]
Permite representar a seguinte configuração do puzzle:

	7	6
5	3	4
1	8	2

2. Representação computacional de um nódo da árvore de procura

Vamos representar um nódo da árvore de procura à custa de um dicionário com as seguintes chaves:

- 'estado' – valor associado será a representação do puzzle correspondente a este nódo;
- 'prog' – dicionário que representa o progenitor deste nódo;
- 'desc' – string que permite descrever o operador que deu origem a este estado;
- 'prof' – valor de profundidade a que o nódo se encontra na árvore de procura;
- 'g' – valor da função de custo para este nódo;
- 'h' – valor da função h para este nódo;
- 'f' – valor da função f para este nódo.

3. Heurísticas

Tal como discutido nas aulas teórico-práticas, vamos adotar 2 heurísticas:

- Soma das distâncias (distância de Manhattan) de cada uma das peças à posição objetivo;

- Número de peças fora do lugar.

1. Define duas funções, `heuristica1` e `heuristica2`, que permitam implementar as duas heurísticas descritas. Assume que as funções recebem como argumentos um nódo da árvore de procura representado como anteriormente descrito.

4. Adaptação dos operadores à representação escolhido para um nódo

1. Reescreve as funções que implementam os quatro operadores assumindo que agora estas recebem como argumento um dicionário com esta estrutura.
   De notar que agora a função deve devolver um novo dicionário com a representação do novo nódo. Será também importante acrescentar um novo teste para verificar se o novo estado é igual ao estado presente no nódo progenitor. Se assim for a função deve devolver False, tal como no caso do movimento não ser possível.

5. Expansão de um nódo

1. Escreve a função `expande` que deve receber um nódo da árvore e devolver uma lista com os filhos deste nódo.

6. Implementação dos algoritmos de procura

1. Escreve a função `procura_em_largura` que deve receber como argumento o nodo inicial de uma instância do problema do puzzle de 8 peças e devolver o nódo correspondente ao estado objectivo.
2. Escreve a função `procura_em_profundidade` que deve receber como argumento o nodo inicial de uma instância do problema do puzzle de 8 peças e devolver o nódo correspondente ao estado objectivo.

Considere a seguinte instância do problema:

```
#Configuração do puzzle que se pretende obter
goal=[[0, 0], [0, 1], [0, 2], # - 1 2
      [1, 0], [1, 1], [1, 2], # 3 4 5
      [2, 0], [2, 1], [2, 2]] # 6 7 8
puzzle_inicial_1 = [[2, 1], [2, 0], [2, 2], # 8 7 6
                    [1, 1], [1, 2], [1, 0], # 5 3 4
                    [0, 2], [0, 1], [0, 0]] # 1 – 2
node_inicial1 = {'estado': puzzle_inicial_1,
```

```
'prog': 'none',
'desc': "",
'prof': 0,
'g': 0,
'h': 22,
'f':22}
```

3. Define uma função que dado o nódo solução, escreve a sequência de operadores que foi necessário aplicar para chegar do estado inicial até ao objetivo.