U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Exactly One Mazes

Inteligência Artificial

Grupo 13_1C:

- André Flores - up201907001
- Diogo Faria - up201907014
- Tiago Rodrigues - up201906807

Definição do Jogo

Encontrar um caminho num tabuleiro entre o Start (canto inferior esquerdo) e o Finish (canto superior direito), em que tem que passar por um e apenas um quadrado de cada peça com formato de L.

Temos como objetivo implementar uma função de pesquisa que consiga resolver tal problema, encontrando a solução ótima para o problema.

Referências: https://erich-friedman.github.io/puzzle/exactly1/

Formulação do Problema como Problema de Pesquisa

- Representação do Estado:
  - Matriz em que: 0 são os espaços vazios, 1 o caminho passado, 2-n (n = Número de L's + 2) as peças em formato de L - Board
  - Posição atual no tabuleiro - Pos = (PosR, PosC)
  - Lista dos Ls a visitar - LVisit
- Estado inicial:
  - Board = Lista de listas com apenas um valor a 1 no canto inferior esquerdo;
  - Pos = (Tamanho da Matriz - 1, 0); Ex: (5, 0)
  - LVisit = Lista com números de 2 até n; Ex: [2, 3, 4, 5, 6, 7]
- Estado Objetivo:
  - Board = Lista de listas com um caminho de 1's até Pos final;
  - Pos = (0, Tamanho da Matriz - 1); Ex: (0, 5)
  - LVisit = Lista vazia; Ex: [ ]
- Heurística:
  - h = distância de Manhattan desde posição corrente até posição final

Ex Inicial:
[[2, 2, 2, 3, 3, 0],
[0, 4, 2, 3, 5, 0],
[0, 4, 0, 3, 5, 6],
[0, 4, 4, 5, 5, 6],
[0, 7, 7, 7, 6, 6],
[1, 0, 0, 7, 0, 0]]

Ex Objetivo:
[[2, 2, 2, 3, 3, 1],
[1, 1, 1, 3, 5, 1],
[1, 4, 1, 1, 1, 1],
[1, 4, 4, 5, 5, 6],
[1, 1, 7, 7, 6, 6],
[1, 1, 0, 7, 0, 0]]

Operadores

Nome	Pré-condições	Pós-Condições	Custo
Down	R < Size-1 && Board[R+1][C] == 0	Pos = (R+1, C) e Board[R+1][C] = 1	1
DownL	R < Size-1 && Board[R+1][C] ∈ LVisit	Pos = (R+1, C), LVisit.remove(Board[R+1][C]) e Board[R+1][C] = 1	1
Up	R > 0 && Board[R-1][C] == 0	Pos = (R-1, C) e Board[R-1][C] = 1	1
UpL	R > 0 && Board[R-1][C] ∈ LVisit	Pos = (R-1, C), LVisit.remove(Board[R-1][C]) e Board[R-1][C] = 1	1
Left	C > 0 && Board[R][C-1] == 0	Pos = (R, C-1) e Board[R][C-1] = 1	1
LeftL	C > 0 && Board[R][C-1] ∈ LVisit	Pos = (R, C-1), LVisit.remove(Board[R][C-1]) e Board[R][C-1] = 1	1
Right	C < Size-1 && Board[R][C+1] == 0	Pos = (R, C+1) e Board[R][C+1] = 1	1
RightL	C < Size-1 && Board[R][C+1] ∈ LVisit	Pos = (R, C+1), LVisit.remove(Board[R][C+1]) e Board[R][C+1] = 1	1

Implementação

- Decidimos usar python para realizar o projeto:
  - Board - lista de listas;
  - LVisit - lista;
  - Pos - tuplo.
- Em termos de ambiente de implementação, utilizamos IDE’s como Visual Studio Code e repositório em GitHub para partilha de código.