# Inteligência Artificial

## Trabalho 1

## Resolução de problemas de pesquisa em espaço de estados


Discentes:

João Santos nº 29634

Rui Rodrigues nº 30388

Évora 2016

# Introdução.

Neste trabalho foi-nos pedido que encarasse-mos o problema do agente que está numa sala de uma caverna que tem nXn salas como um problema de pesquisa em espaços de estados.  A condição implica que as casas (1,2) e (1,3), (2,3) e (2,2), (3,4) e (4,4), e (4,5) e (3,5) estão bloqueadas e a travessia deve ser feita da casa (2,2) para a (8,8).

# 1 – Respostas.

## 1.

Para encararmos o problema como espaço de estados vai ser necessário definir os estados e os operadores de transição.

Os estados são definidos com um tuplo com um par ordenado e uma lista, o par ordenado para indicar as coordenadas no tabuleiro e a lista para guardar os estados visitados.

O estado inicial é definido com a casa inicial.

```
estado_inicial(c((2,2), [])).
```

O estado final é definido com a casa de destino.

```
estado_final(c((8,8), _)).
```

Para os operadores das transições definimos 4 movimentos: cima, baixo, esquerda e direita. A transição é feita incrementando/decrementando as coordenadas X e Y.

```
op(c((X,Y), L), sobe, c((X,Z), [(X,Y)|L]), 1):-
        Z is Y+1,
        Z < 31,
        restricoes(Rest),
        \+ member((X,Z), Rest),
        \+ member((X,Z), L).
```

```
op(c((X,Y), L), desce, c((X,Z), [(X,Y)|L]), 1):-
                    Z is Y-1,
                    Z > -1,
                    restricoes(Rest),
                    \+ member((X,Z), Rest),
                    \+ member((X,Z), L).

op(c((X,Y), L), esquerda, c((Z,Y), [(X,Y)|L]), 1):-
                    Z is X-1,
                    Z > -1,
                    restricoes(Rest),
                    \+ member((Z,Y), Rest),
                    \+ member((Z,Y), L).

op(c((X,Y), L), direita, c((Z,Y), [(X,Y)|L]), 1):-
                    Z is X+1,
                    Z < 31,
                    restricoes(Rest),
                    \+ member((Z,Y), Rest),
                    \+ member((Z,Y), L).
```

## 1.(a) e (b)

O código de pesquisa que resolve este problema devolvendo sempre a melhor solução é um código de pesquisa em largura. Para a resolução criada utilizamos o código fornecido pela docente contido no ficheiro pni.pl, dado que a pesquisa não informada para ir da sala (18,18) para a sala (26,26) utiliza a stack toda (utilizámos 2GB de global stack), não nos foi possível concluír quantos nós são visitados para chegar lá. Uma solução seria guardar os nós já visitados mas isso seria dar-lhe informação, logo utilizámos isso na pesquisa informada.

## 2.(a)

Utilizámos duas heurísticas, a norma (raíz das coordenadas ao quadrado) e a soma da distancia absoluta das coordenadas X e Y. Comentámos à vez para utilizar no ficheiro pi.pl as heurísticas.

```
h(c((X,Y), _), H):- estado_final(c((W,Z), _)),
                    CoordX is abs(W-X),
                    CoordY is abs(Z-Y),
                    H is CoordY + CoordX.


%h(c((X,Y), _), H):- estado_final(c((W,Z), _)),
%                    CoordX is abs(W-X),
%                   CoordY is abs(Z-Y),
%                  H is round(sqrt(CoordY*CoordY + CoordX*CoordX)).
```

## 2.(b)

O código utilizado fornecido pela docente contido no ficheiro pi.pl foi o utilizado para resolver este problema, alterámos o código de forma a que fosse guardado em memória os estados já visitados de forma a que não fossem expandidos e a pesquisa não andasse “às voltas” sobre o mesmo.

## 2.(c)

Obtivemos o seguinte output:

```
custo(16)
profundidade(16)
80
```

O número 80 reprenta o número exacto de nós visitados e dado que decidímos não voltar a estados repetidos, isto é, expandí-los, o número máximo também será 80.