UNIVERSIDADE DE ÉVORA

# Inteligência Artificial

## Trabalho 1


Miguel Grilo 58387

Jorge Couto 58656

Colégio Luís António Verney

# Tabela de Conteúdos

# Exercício 1

## Alínea a)

### Exemplo 1

- $\% \ e((I,J),[Itens]) \ \% \ Uma \ lista \ que \ escreve \ o \ estado \ de \ cada \ casa.$

  $\% \ Primeira \ casa: posição \ em \ linha \ (i) \ e \ coluna \ (j).$

  $\% \ Segunda \ casa: Lista \ para \ acumular \ os \ itens.$

- $estado\_inicial(e(c(1,1),[])).$
- $Elementos \ Estáticos \ (posições \ dos \ 'X'):$
  - $x(2,1).$
  - $x(6,1).$
  - $x(1,4).$
  - $x(2,4).$
  - $x(3,4).$
  - $x(4,4).$
  - $x(5,3).$
  - $x(6,3).$
  - $x(7,4).$
  - $x(6,7).$
- $Objetos \ (posições \ das \ Chaves):$
  - $objeto(a,4,2).$
  - $objeto(b,6,2).$
- $estado\_final(e(c(7,2),[b,a])).$

### Exemplo 2

- $estado\_inicial2(e(c(1,1),[])).$
- $Elementos \ Estáticos \ (posições \ dos \ 'X'):$
  - $x(6,1).$
  - $x(6,3).$
  - $x(2,4).$
  - $x(3,4).$
  - $x(6,7).$
  - $x(5,5).$
- $Objetos \ (posições \ das \ Chaves):$
  - $objeto(a,7,3).$
  - $objeto(b,4,3).$
- $estado\_final2(e(c(7,5),[b,a])).$

### Alínea b)

- $valido(I,J) :-$
  $I >= 1,$
  $I =< 7,$
  $J >= 1,$
  $J =< 7,$
  $\backslash + x(I,J).$
- $move((X,Y),(NX,Y)) :- NX\ is\ X + 1. \%\ direita$
- $move((X,Y),(NX,Y)) :- NX\ is\ X - 1. \%\ esquerda$
- $move((X,Y),(X,NY)) :- NY\ is\ Y + 1. \%\ cima$
- $move((X,Y),(X,NY)) :- NY\ is\ Y - 1. \%\ baixo$

- $op(e(c(I,J),Itens), anda, e(c(INovo,JNovo),Itens),1) :-$
  $move((I,J),(INovo,JNovo)),$
  $valido(INovo,JNovo).$
- $op(e(c(I,J),Itens), apanhar, e(c(I,J),[a|Itens]),1) :-$
  $objeto(a,I,J),$
  $\backslash + member(a,Itens).$
- $op(e(c(I,J),Itens), apanhar, e(c(I,J),[b|Itens]),1) :-$
  $objeto(b,I,J),$
  $member(a,Itens),$
  $\backslash + member(b,Itens).$

### Alínea c)

Consultamos o ficheiro [pesquisa_nao_informada] e [robot] para resolver o problema.

Tendo *b* como máximo factor de ramificação da árvore de pesquisa, *d* como profundidade da solução de menor custo e *m* como profundidade máxima do espaço de estados.

#### Exemplo 1

**Pesquisa em Largura ( $\boldsymbol{pesquisa('robot', largura).}$ ):**

- Custo: 9
- Profundidade: 9
- Nós visitados (estimados) : $b^{d+1} = 3^{10} = 59049$
- Nós visitados (concretos) : Não determinado devido a stack overflow.
- Nós em memória (estimados) : $b^{d+1} = 3^{10} = 59049$
- Nós em memória (concretos) : Não determinado devido a stack overflow.

**Pesquisa em Profundidade ( $\boldsymbol{pesquisa(\text{'}robot\text{'}, profundidade).}$ ):**

- Custo: 9
- Profundidade: 9
- Nós visitados (estimados) : $b^m = 3^{114} \approx 2.5 \times 10^{54}$
$$m = (\mathbf{7} \times \mathbf{7}\ posições\ -\ 11\ 'x') \times\ \mathbf{3}\ combinações\ de\ objetos = \mathbf{114}$$
- Nós visitados (concretos) : Não determinado devido a stack overflow.
- Nós em memória (estimados) : $b.m = \ 3 \times 114 = 342$
- Nós em memória (concretos) : Não determinado devido a stack overflow.

**Pesquisa em Profundidade Iterativa ( $\boldsymbol{pesquisa(\text{'}robot\text{'}, profIt).}$ ):**

- Custo: 9
- Profundidade: 9
- Nós visitados (estimados) : $b^d = 3^9 = 19683$
- Nós visitados (concretos) : 3503
- Nós em memória (estimados) : $b.d = \ 3 \times 9 = 27$
- Nós em memória (concretos) : 24

**Pesquisa em Profundidade Limitada (9) ( $\boldsymbol{pesquisa(\text{'}robot\text{'}, profLim(9)).}$ ):**

- Custo: 9
- Profundidade: 9
- Nós visitados (estimados) : $b^l = 3^9 = 19683$
- Nós visitados (concretos) : 579
- Nós em memória (estimados) : $b.l = \ 3 \times 9 = 27$
- Nós em memória (concretos) : 24

## Exemplo 2

**Pesquisa em Largura ( $\boldsymbol{pesquisa(\text{'}robot\text{'}, largura).}$ ):**

- Custo: 20
- Profundidade: 20
- Nós visitados (estimados) : $b^{d+1} = 3^{21} \approx 1 \times 10^{10}$
- Nós visitados (concretos) : Não determinado devido a stack overflow.
- Nós em memória (estimados) : $b^{d+1} = 3^{21} \approx 1 \times 10^{10}$

- Nós em memória (concretos) : Não determinado devido a stack overflow.

**Pesquisa em Profundidade ( $\boldsymbol{pesquisa(\text{‘}robot\text{’}, profundidade).}$ ):**

- Custo: 20
- Profundidade: 20
- Nós visitados (estimados) : $b^m = 3^{129} \approx 3.5 \times 10^{61}$
  $m = (\mathbf{7} \times \mathbf{7}\ posições - 6\ 'x') \times\ \mathbf{3}\ combinações\ de\ objetos = \mathbf{129}$
- Nós visitados (concretos) : Não determinado devido a stack overflow.
- Nós em memória (estimados) : $b.m = 3 \times 129 = 387$
- Nós em memória (concretos) : Não determinado devido a stack overflow.

**Pesquisa em Profundidade Iterativa ( $\boldsymbol{pesquisa(\text{‘}robot\text{’}, profIt).}$ ):**

- Custo: 20
- Profundidade: 20
- Nós visitados (estimados) : $b^d = 3^{20} \approx 3.5 \times 10^9$
- Nós visitados (concretos) : Não determinado devido a stack overflow.
- Nós em memória (estimados) : $b.d = 3 \times 20 = 60$
- Nós em memória (concretos) : Não determinado devido a stack overflow.

**Pesquisa em Profundidade Limitada (9) ( $\boldsymbol{pesquisa(\text{‘}robot\text{’}, profLim(9)).}$ ):**

- Custo: 20
- Profundidade: 20
- Nós visitados (estimados) : $b^l = 3^{20} \approx 3.5 \times 10^9$
- Nós visitados (concretos) : Não determinado devido a stack overflow.
- Nós em memória (estimados) : $b.l = 3 \times 20 = 60$
- Nós em memória (concretos) : Não determinado devido a stack overflow.

Assim, concluímos que o algoritmo de pesquisa não informada mais eficiente para resolver este problema é a Pesquisa em Profundidade Limitada, que encontra a solução utilizando menos memória e menos tempo de computação em comparação com as restantes pesquisas não informadas.

- $pesquisa\_pLim([no(E, Pai, Op, C, P)|\_], no(E, Pai, Op, C, P), \_) :-$
  $estado\_final(E), inc.$
- $pesquisa\_pLim([E|R], Sol, Pl) :- inc, expandePl(E, Lseg, Pl),$
  $insere\_fim(R, Lseg, Resto),$
  $length(Resto, N), actmax(N),$
  $pesquisa\_pLim(Resto, Sol, Pl).$

## Alínea d)

Consultamos o ficheiro [pesquisa_nao_informada] e [robot] para resolver o problema.

### Exemplo 1

$\boldsymbol{pesquisa}(\text{‘}\boldsymbol{robot}\text{’}, \boldsymbol{profLim}(\mathbf{9})).$

*Ponto i.*

Número total de espaços visitados: 579

*Ponto ii.*

Número de espaços simultaneamente em memória: 24

### Exemplo 2

$\boldsymbol{pesquisa}(\text{‘}\boldsymbol{robot}\text{’}, \boldsymbol{profLim}(\mathbf{9})).$

*Ponto i.*

Número total de espaços visitados: Não determinado devido a stack overflow.

*Ponto ii.*

Número de espaços simultaneamente em memória: Não determinado devido a stack overflow.

## Alínea e)

$$heurística_1(s) = d(s, A) + d(A, B) + d(B, \text{saída})$$

$$heurística_2(s) = d(s, \text{saída}) + 2 - |\text{Itens Apanhados}|$$

**Código em Prolog:**

- $\%\ Distância\ Manhattan\ entre\ duas\ posições\ c(I, J)$
  $distancia(c(X1, Y1), c(X2, Y2), D)\ :-$
  $DX\ is\ abs(X1\ -\ X2),$
  $DY\ is\ abs(Y1\ -\ Y2),$
  $D\ is\ DX\ +\ DY.$

- $\%Heurística\_1$
  $heuristica\_h1(e(Pos, Itens), H) :-$
  $PosA = c(4,2), PosB = c(6,2), PosSaida = c(7,2),$
  $\%\ Trocar\ entre\ c(7,2)\ ou\ c(7,5)\ conforme\ o$
  $exemplo\ 1\ ou\ 2\ respectivamente$
  $(member(a, Itens) -> DA = 0\ ;\ distancia(Pos, PosA, DA)),$

  $(member(b, Itens) -> DB = 0, DS$
  $= 0;\ distancia(PosA, PosB, DB), distancia(PosB, PosSaida, DS)),$

  $H\ is\ DA + DB + DS.$
- $\%Heurística\_2$
  $heuristica\_h2(e(PosAtual, Itens), H) :-$
  $PosSaida = c(7,2),$
  $distancia(PosAtual, PosSaida, D),$
  $length(Itens, N),$
  $H\ is\ D + 2 - N.$

## Alínea f)

- A* é a pesquisa informada que combina custo já gasto (g) e estimativa do custo restante (h) para garantir encontrar a solução ótima de forma eficiente, logo foi escolhida como melhor algoritmo de pesquisa informada para este problema.
- $\%\ AlgoritmoA^*$
  $pesquisa\_a\big([no(E, Pai, Op, C, HC, P)|\ \ ], no(E, Pai, Op, C, HC, P)\big):-$

  $estado\_final(E).$

  $pesquisa\_a([E|R], Sol):- expandea(E, Lseg), \%esc(E),$
  $insere\_ord(Lseg, R, Resto),$
  $pesquisa\_a(Resto, Sol).$

- A heuristica_h1 é mais informativa e mais precisa, pois soma as distâncias até apanhar os objetos na ordem correta e depois até à saída, refletindo melhor o custo real esperado. Assim, reduz a expansão desnecessária de estados e guia o algoritmo melhor que a heurística_h2, sendo, portanto, a heurística_h1 escolhida como melhor heurística para este problema.
- $\%Heurística\_1$
  $heuristica\_h1(e(Pos, Itens), H) :-$
  $PosA = c(4,2), PosB = c(6,2), PosSaida = c(7,2),$

  $\%\ Trocar\ entre\ c(7,2)\ ou\ c(7,5)\ conforme\ o$

  $exemplo\ 1\ ou\ 2\ respectivamente$

$(member(a, Itens) -> DA = 0 ; distancia(Pos, PosA, DA)),$

$(member(b, Itens) -> DB = 0, DS = 0; distancia(PosA, PosB, DB), distancia(PosB, PosSaida, DS)),$

$H \; is \; DA + DB + DS.$

## Alínea g)

Consultamos o ficheiro [pesquisa_informada] e [robot] para resolver o problema.

### Exemplo 1

$\boldsymbol{pesquisa}(‘\boldsymbol{robot}’, \boldsymbol{a}).$

*Ponto i.*

Número total de estados visitados: 24

*Ponto ii.*

Número de estados simultaneamente em memória: 15

### Exemplo 2

$\boldsymbol{pesquisa}(‘\boldsymbol{robot}’, \boldsymbol{a}).$

*Ponto i.*

Número total de estados visitados: 306

*Ponto ii.*

Número de estados simultaneamente em memória: 156

## Soluções Ótimas

### Exemplo 1

$e([], e(c(1,1), []))$

$e(anda, e(c(1,2), []))$

$e(anda, e(c(2,2), []))$

$e(anda, e(c(3,2), []))$

$e(anda, e(c(4,2), []))$

$e(apanhar, e(c(4,2), [a]))$

$e(anda, e(c(5,2), [a]))$

$e(anda, e(c(6,2), [a]))$

$e(apanhar, e(c(6,2), [b, a]))$

$e(anda, e(c(7,2), [b, a]))$

## Exemplo 2

$e([], e(c(1,1), []))$

$e(anda, e(c(2,1), []))$

$e(anda, e(c(3,1), []))$

$e(anda, e(c(4,1), []))$

$e(anda, e(c(4,2), []))$

$e(anda, e(c(5,2), []))$

$e(anda, e(c(6,2), []))$

$e(anda, e(c(7,2), []))$

$e(anda, e(c(7,3), []))$

$e(apanhar, e(c(7,3), [a]))$

$e(anda, e(c(7,4), [a]))$

$e(anda, e(c(6,4), [a]))$

$e(anda, e(c(5,4), [a]))$

$e(anda, e(c(4,4), [a]))$

$e(anda, e(c(4,3), [a]))$

$e(apanhar, e(c(4,3), [b, a]))$

$e(anda, e(c(5,3), [b, a]))$

$e(anda, e(c(5,4), [b, a]))$

$e(anda, e(c(6,4), [b, a]))$

$e(anda, e(c(7,4), [b, a]))$

$e(anda, e(c(7,5), [b, a]))$