

1º Trabalho Prático

Resolução de Problemas

Licenciatura em Eng. Informática
Inteligência Artificial



Helder Godinho 42741

Mariana Silva 54389

Docente: Irene Pimenta

1. Considere o seguinte problema: Um Agente A tem como objectivo colocar uma máquina M na saída S de- pois de apanhar os objectos (os), o mais rapidamente possível. O Agente pode mover-se para cima, baixo, esquerda e direita mas não pode ir para as casas marcadas com um x. A máquina M pode ser empurrada pelo Agente e quando está numa casa com um objeto se o agente estiver numa casa adjacente pode acionar uma alavanca para a máquina apanhar o objecto.

1º exemplo:

·	1	2	3	4	5	6	7
7		S		x			
6	x		x				x
5							
4		o		x			
3				x			
2		M					
1		A					

2º exemplo:

·	1	2	3	4	5	6	7
7			x		S		
6	x		x				x
5					o		
4			x				
3				x			
2		M		x			
1		A					

3º exemplo:

·	1	2	3	4	5	6	7
7			x		S		
6	x		x				x
5							
4		o		x			
3				x			
2		M				o	
1		A					

a) Represente em Prolog o estado inicial e o estado final para cada um dos 3 exemplos.

```
1      %e(Agente, Maquina, CasasBloqueadas, Objetos)
2
3      %Exemplo 1
4      %estado_inicial(e(a(1,2), m(2,2), [(6,1), (7, 4), (6, 3), (7,4), (4, 4), (3,
5          4), (2, 4), (6, 7)], [(4,2)])).
6      %estado_final(e(_, m(7, 2), _, [])).
7
8      %Exemplo 2
9      %estado_inicial(e(a(1,2), m(2,2), [(6,1), (7, 3), (6, 3), (3,4), (4, 3), (3,
10         4), (2, 4), (6, 7)], [(5,5)])).
11      %estado_final(e(_, m(7,5), _, [])).
12
13      %Exemplo 3
14      %estado_inicial(e(a(1,2), m(2,2), [(6,1), (7, 4), (7, 3), (6,3), (4, 4), (3,
15         4), (2, 4), (6, 7)], [(4,2), (6,2)])).
16      %estado_final(e(_, m(7,5), _, [])).
```

b) Represente em Prolog os operadores de transição de estados para este problema.

```
1      %Movimentos possiveis
2      move([(0,1), (0, -1), (1, 0), (-1, 0)]).
3
4      %Operação mover agente e maquina se tiver na mesma direção
5      op(e(a(X, Y), m(Xm, Ym), CB, Os), (L, C), e(a(X1, Y1), m(Xm1, Ym1), CB, Os),
6          1) :-
7          move(M),
8          member((L, C), M),
9          X1 is X + L,
10         Y1 is Y + C,
11         X1 > 0, X1 < 8,
12         Y1 > 0, Y1 < 8,
13         \+ member((X1, Y1), CB),
14         (Xm = X1, Ym = Y1 -> (Xm1 is Xm + L, Ym1 is Ym + C); Xm1 = Xm, Ym1 = Ym)
15         .
16
17      %Operação ativar alavanca
18      op(e(a(X, Y), m(Xm, Ym), CB, Os), ativa_alavanca, e(a(X, Y), m(Xm, Ym), CB,
19          Os1), 1) :-
20          member((Xm, Ym), Os),
21          delete(Os, (Xm, Ym), Os1).
```

c) Represente o código em Prolog do algoritmo de pesquisa não informada mais eficiente a resolver este problema. Para justificar a escolha do algoritmos deve apresentar o número uma estimativa do número de nós visitados e em memória para cada algoritmo..

O algoritmo de pesquisa mais eficiente a resolver este problema é o **Algoritmo de pesquisa em largura**.

Algoritmo de pesquisa em largura

Como estamos perante um problema que irá ter em média um fator de ramificação, b, igual a 5, pois o agente pode mover-se nas quatro direções e ainda ativar a alavanca, iremos ter bastantes

nós em memória. Também sabemos que a nossa solução no primeiro exemplo está a 6 estados da solução (cima, cima, ativar alavanca, cima, cima, cima).

$$\text{Logo } N = 5^1 + 5^2 + 5^3 + 5^4 + 5^5 + 5^6 + 5^7 = 97656$$

Onde N representa o número máximo de nós que podem ser visitados pelo algoritmo de pesquisa em largura.

Código do algoritmo de pesquisa em largura:

```
1  % Busca em largura (BFS)
2  bfs(Estado, Caminho, NVisitados) :-
3      bfs([[Estado]], [], Caminho, 0, NVisitados).
4
5  bfs([[Estado|Caminho]|_], _, [Estado|Caminho], NVisitados, NVisitados) :-
6      estado_final(Estado).
7
8  bfs([[Estado|Caminho]|OutrosCaminhos], Visitados, Solucao, NVisAnt,
9      NVisitados) :-
10     findall([NovoEstado, Estado|Caminho],
11             (op(Estado, _, NovoEstado, _), \+ member(NovoEstado, Visitados)),
12             NovosCaminhos),
13     append(OutrosCaminhos, NovosCaminhos, TodosCaminhos),
14     length(OutrosCaminhos, N),
15     NVisSeg is NVisAnt + N,
16     bfs(TodosCaminhos, [Estado|Visitados], Solucao, NVisSeg, NVisitados).
```

Algoritmo de pesquisa em profundidade:

Tentámos utilizar o algoritmo de pesquisa em profundidade mas nunca obtivemos a solução. Um dos principais motivos pelos quais o algoritmo de pesquisa em profundidade pode não encontrar uma solução é quando o algoritmo entra em um ciclo de estados em que continua a explorar caminhos sem chegar a um estado final, entrando assim num ciclo infinito.

c) Depois de resolver os 3 exemplos deste problema com o algoritmo da alínea anterior indique:

i) qual o número total (exacto) de estados visitados

Exemplo 1: $N = 48725$

Exemplo 2: $N = 2199974$

Exemplo 3: Global stack overflow

ii) qual o máximo número (exacto) de estados que têm que estar simultaneamente em memória.

$$N_{\text{memoria}} = b * (d + 1)$$

Onde:

- N_{memoria} é o número máximo de estados em memória.
- fator_ramificacao (b) é o número máximo de sucessores gerados por cada estado.
- d é a profundidade máxima da árvore de busca.

Exemplo 1: $d = 6, N_{\text{memoria}} \leq 35$

Exemplo 2: $d = 13, N_{\text{memoria}} \leq 70$

Exemplo 3: $d = 29$, $N_{\text{memoria}} \leq 150$

e) Proponha duas heurísticas admissíveis para estimar o custo de um estado até à solução para este problema.

Distância de Manhattan: Também conhecida como métrica L1 ou distância retangular, calcula a distância como a soma das diferenças absolutas entre as coordenadas dos pontos ao longo de cada dimensão.

Distância euclidiana: Também conhecida como métrica L2, é a linha reta mais curta entre dois pontos em um espaço euclidiano. É calculada usando o teorema de Pitágoras para encontrar a hipotenusa de um triângulo retângulo formado pelas diferenças nas coordenadas dos pontos.

f) Apresente o código em Prolog do algoritmo de pesquisa informada mais eficiente para resolver este problema usando as heurísticas definidas na alínea anterior. Justifique a escolha do melhor algoritmo e heurística.

O algoritmo A* com a heurística da distância de Manhattan foi escolhido como a melhor opção para resolver o problema. Ele oferece uma solução ótima e completa, garantindo eficiência na busca. Além disso, a heurística da distância de Manhattan é adequada para este problema, permitindo ao algoritmo fazer escolhas inteligentes ao longo do caminho. Em contraste, o Greedy é menos eficiente. Assim, o A* com a heurística da distância de Manhattan se destacou como a escolha mais adequada.

Algoritmo a*

```

1  pesquisa_a([no(E,Pai,Op,C,HC,P)|_],no(E,Pai,Op,C,HC,P)):-
2      estado_final(E),
3      inc.
4
5  pesquisa_a([E|R],Sol):- inc,
6      asserta(fechado(E)),
7      expande(E,Lseg),
8      esc(E),
9      insere_ord(Lseg,R,Resto),
10     length(Resto,N), actmax(N),
11     pesquisa_a(Resto,Sol).
12
13 inc:- retract(nos(N)), N1 is N+1, asserta(nos(N1)).
14
15 expande(no(E,Pai,Op,C,HC,P),L):-
16     findall(no(En,no(E,Pai,Op,C,HC,P),Opn,Cnn,HCnn,P1),
17             (op(E,Opn,En,Cn), \+ fechado(no(En,_,_,_,_,_))),
18             P1 is P+1, Cnn is Cn+C, h(En,H), HCnn is Cnn
19             +H), L).
20
21 insere_ord([],L,L).
22 insere_ord([A|L],L1,L2):- insereE_ord(A,L1,L3), insere_ord(L,L3,L2).
23
24 insereE_ord(A,[],[A]).
25 insereE_ord(A,[A1|L],[A,A1|L]):- menor_no(A,A1),!.
26 insereE_ord(A,[A1|L],[A1|R]):- insereE_ord(A,L,R).
27
28 menor_no(no(_,_,_,_,N,_), no(_,_,_,_,N1,_)):- N < N1.

```

g) Depois de resolver os 3 exemplos deste problema com o algoritmo da alínea anterior indique para cada função heurística:

i) qual o número total (exacto) de estados visitados

Resultados com heurística Distância de Manhattan:

1º exemplo: Nós visitados: 25

2º exemplo: Nós visitados: 362

3º exemplo: Nós visitados: Sem resultados

Resultados com heurística Distância euclidiana:

1º exemplo: Nós visitados: 25

2º exemplo: Nós visitados: 523

3º exemplo: Nós visitados: Sem resultados

ii) qual o máximo número (exacto) de estados que têm que estar simultaneamente em memória.

Resultados com heurística Distância de Manhattan:

1º exemplo: Nós em memória: 46

2º exemplo: Nós em memória: 246

3º exemplo: Nós em memória: Sem resultados

Resultados com heurística Distância euclidiana:

1º exemplo: Nós em memória: 46

2º exemplo: Nós em memória: 337

3º exemplo: Nós em memória: Sem resultados