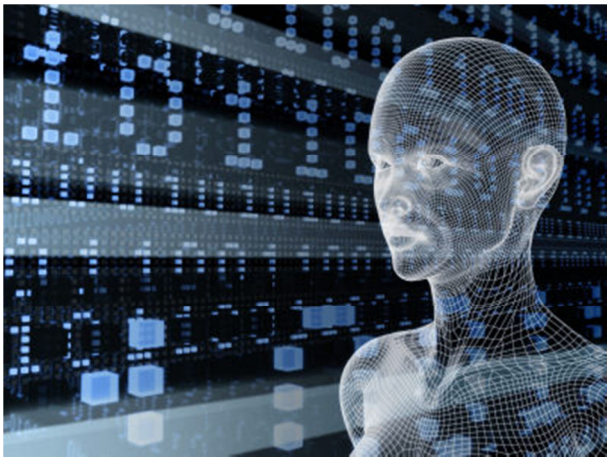

1º Trabalho Prático

Trabalho Elaborado Por:

Luís Polha Nº20464

Marlene Oliveira Nº 25999

Ano Lectivo 2011/2012

1.

A representação em Prolog utilizada para o espaço de estados e operadores de transição de estados para este problema pode ser consultada no ficheiro trabalho.pl.

a. O algoritmo de pesquisa não informada mais eficiente para resolver este problema será o algoritmo de Pesquisa em Profundidade Iterativa.

Este algoritmo expande o nó mais profundo que ainda não foi expandido, sendo limitada a profundidade máxima a que os nós podem ser expandidos. Se não for encontrada uma solução, a profundidade a que a pesquisa estava limitada aumenta e o processo repete-se.

Código Prolog do algoritmo:

```
iterativa(ListaEstados,Solucao,Prof):-
        limitada(ListaEstados,Solucao,Prof).
iterativa(ListaEstados,Solucao,Prof):-
        PNext is Prof+1,
        limitada(ListaEstados,Solucao,PNext).


limitada([E|R],S,Pl):-
        expande_pni(E,Next,Pl),
        inserir(R,Next,Resto),
        limitada(Resto,S,Pl).


limitada([no(E,Pai,Opera,Cust,Prof)|_],no(E,Pai,Opera,Cust,Prof),_):-
        estado_final(E).


expande_pni(no(EstAct,Pai,Opera,Cust,Prof),Lista,_):-
        findall(no(EstSeg,no(EstAct,Pai,Opera,Cust,Prof),OperaSeg,CustoTotal,P1),(op(EstAct,OperaSeg,EstSeg,CustSeg),
                P1 is Prof+1,CustoTotal is CustSeg+Cust),Lista).
```

Para o caso em que a caverna tem 30x30 salas, este será também o algoritmo mais eficiente, uma vez que não expandirá todos os nós como acontece com a pesquisa em largura e, como é um algoritmo completo, irá sempre encontrar uma solução.

b.

i. Sabemos que a complexidade temporal para o algoritmo de pesquisa em profundidade iterativa é $O(b^d)$. Como o *branching factor* (b) da árvore será 4 (cada estado expandido terá 4 filhos) e a profundidade da solução de menor custo (d) será 16, sabemos que o número de estados visitados, no pior caso, será 4^{16} = 4294967296 estados.

ii. Sabemos que a complexidade espacial (número máximo de estados em memória) para o algoritmo de pesquisa em profundidade iterativa é O(bd). Então, o número máximo de estados em memória, no pior caso, será 4*16 = 64 estados.

2.

a. Para estimar o custo de um estado até à solução para o problema apresentado, propõem-se as seguintes heurísticas:

- Distância de Manhattan: número de salas até à sala correcta.
  - Código Prolog desta heurística:

```
distancia_m((X,Y),D) :-
        estado_final((W,Z)),
        D is abs(X-W)+abs(Y-Z).
```

- Distância Euclidiana: Simplificação do problema, de modo a que se possam contar as salas, na diagonal, até à sala correcta.
  - Código Prolog desta heurística:

```
distancia((X,Y),D) :-

        estado_final((W,Z)),

        D is round(sqrt(abs(X-W)^2+abs(Y-Z)^2)).
```

b. O algoritmo de pesquisa informada mais eficaz para a resolução deste problema será o A*, uma vez que este algoritmo evita os caminhos que possuem custos mais elevados (utilizando uma função de avaliação que soma o custo até atingir a solução com o custo estimado até atingir essa mesma solução utilizando a heurística), ou seja, expande os nós que possuem menor custo até que uma solução seja encontrada.

Código Prolog do algoritmo:

```
a([no(E,Pai,Op,C,HC,P)|_],no(E,Pai,Op,C,HC,P)):- estado_final(E).

a([EstA|Restantes],Sol):-

        expande_i(EstA,Seguintes),
```

```
    ordem(Seguintes,Restantes,Ordenados),
    a(Ordenados,Sol).

expande_i(no(EA,Pai,OpA,CA,HA,PA),Lista):-
    findall(no(ES,no(EA,Pai,OpA,CA,HA,PA),OpS,CTotal,HTotal,PS),
        ( op(EA,OpS,ES,CS),
        PS is PA+1,
        CTotal is CA+CS,
        distancia_m(EA,HS),
        HTotal is HA+HS),
      Lista).
```

c.

i. São visitados 48 estados com ambas as heurísticas.

ii. Em memória, em ambos os casos, encontrar-se-ão 12 estados.

3.

a. Para a representação dos estados decidimos admitir caminhos que não são soluções possíveis de maneira a facilitar a geração de estados iniciais e vizinhos. Assim a nossa representação de caminhos vai ser algo semelhante a:

estado ([estado_inicial, outras_posições.....,estado_final]).

Exemplo da representação de um estado utilizando a nossa notação:

estado([(2,2),(2,3),(3,3),(8,8)]).

b. Nesta alínea não conseguimos resolver o predicado vizinho porque não conseguimos pensar numa maneira para percorrer o caminho até ao "buraco", depois calcular as casas vizinhas e inserir cada um dos elementos da lista das casas vizinhas na lista original para assim retornar uma lista de listas com os vizinhos.

c. Como não conseguimos resolver a alínea anterior não conseguimos fazer esta.