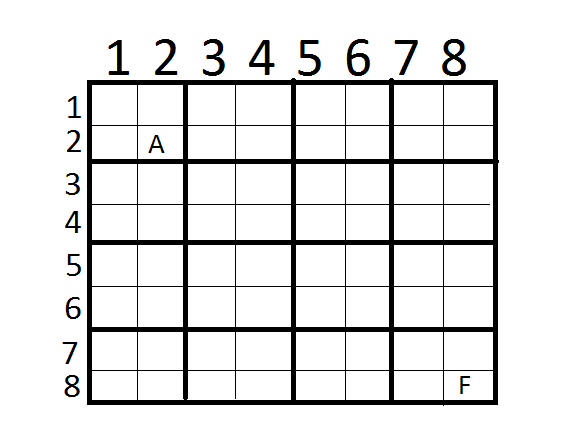
Inteligência Artificial

Trabalho 2015/2016

16/03/16





Professora:

Irene Pimenta Rodrigues

Realizado por:

João Calhau - 31621

José Pimenta - 31677

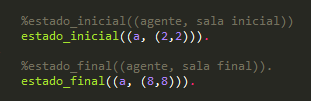
Introdução:

Este trabalho enquadra-se na disciplina de Inteligência Artificial e vamos abordar pesquisas não informadas e pesquisas informadas para resolução de um exercicio definido pela professora, em que temos de pesquisar um caminho numa matriz NxN de um dado ponto até chegar a outro ponto.

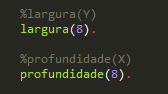
Iremos assim tentar dar o nosso melhor, e iremos tentar utilizar os métodos que achemos mais correctos ou propícios à boa evolução do trabalho e que no final se concretize o que nos é pedido.

**Exercicio 1:**

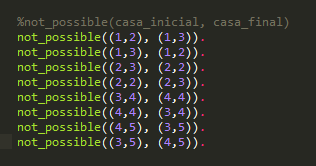
Para representar o espaço de estados decidimos fazê-lo utilizado 1 tuplo para o estado inicial que contém a letra do agente e a sala de partida(que é 1 tuplo com a posição X e Y), e o mesmo se aplica ao estado final:



Para ajudar nas operações e definir o tamanho da matriz usamos ainda umas variáveis estáticas que definem a largura e profundidade:



Para realizar o problema temos ainda as transições que não se podem realizar, ou seja, as portas bloqueadas (transição em ambos os sentidos):



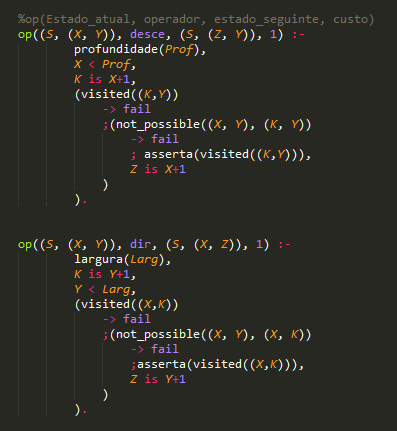
Para a resolução do problema temos ainda uma definição dynamic Visited que regista os nós que já foram passados de modo a evitar loops:

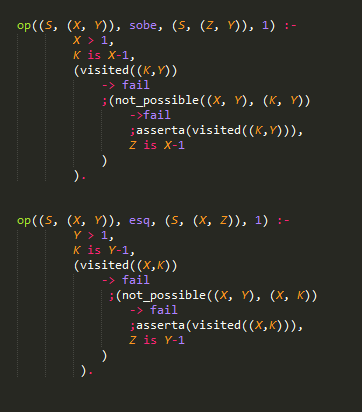


Para as operações, temos 4 que são andar para baixo, direita, cima e esquerda.

Com base no nó em que se encontra e conforme a operação verifica-se se o nó a que se chega com essa operação é um nó que ainda não foi visitado ou se é um nó que não tenha porta bloqueante.

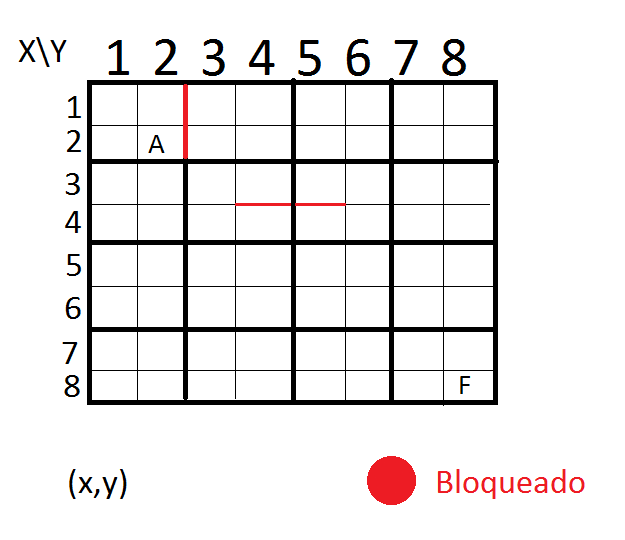
As operações estão definidas então da seguinte maneira:





**Pergunta 1 a)**

Usando como exercicio o definido em cima de uma sala 8x8 e começando o agente na sala (2,2) e tentando ir para a sala (8,8) tendo as portas bloqueadas entre (1,2) e (1,3), (2,3) e (2,2), (3,4) e (4,4), e (4,5) e (3,5), um desenho representativo seria o seguinte:



Tendo em conta que guardamos os locais por onde passamos, em termos de largura o algoritmo vai expandindo como um “ripple”, propagando-se pelos vizinhos em redor. Logo em termos de largura o algoritmo ia funcionar algo como:



Assim e sabendo que são guardados nós por onde já se passou, no pior dos casos corre-se a matriz toda que será N\*N, neste caso 8\*8 = 64 nós + 1 que é o nó inicial para o qual tenta ir (pois este inicialmente não está registado como já visitado).

Sendo assim se o estado final se encontrar a uma distância tanto no X como no Y maior que metade de N ( ou seja, neste caso se a diferença entre o estado inicial fosse >= a 4, em principio todos os nós seriam preenchidos através do algoritmo).

Numa pesquisa de profundidade, e tendo em a ordem das nossas operações em conta, sendo a ordem descer, direita, cima, esquerda, o algoritmo vai descendo nas posiçoes até não dar mais, e depois vai para a direita e atinge facilmente a posição final. Mas caso a ordem fosse por exemplo esquerda, cima, direita, descer, o algoritmo demora mais tempo a resolver.

Assim sendo, quando se verifica que estado\_inicial(a,b) e estado\_final (c,d), e que c-a >= N/2 e que d-b >= N/2, deve-se utilizar a pesquisa em profundidade.

Em caso contrário, o estado final está relativamente perto do estado inicial e assim uma pesquisa em largura é mais rápida.

A pesquisa iterativa não funciona assim no nosso trabalho, devido a que na 1ª iteração guardamos logo os nós vizinhos, e na 2ª iteração não lhes vai conseguir aceder, é assim um compromisso para os outros algoritmos funcionarem de modo muito mais rápido.

**Pergunta 1 b)**

LARGURA:

Número de nós visitados: 65

Número máximo de nós em memória: 10

PROFUNDIDADE:

Número de nós visitados: 13

Número máximo de nós em memória: 19

(Ordem de operações definidas no nosso trabalho: descer, direita, cima, esquerda)

PROFUNDIDADE:

Número de nós visitados: 44

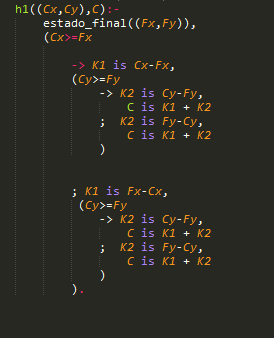
Número máximo de nós em memória: 22

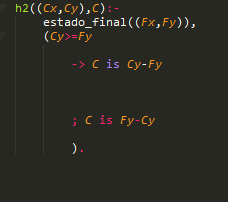
(Ordem de operações: esquerda, cima, direita, descer)

**Pergunta 2 a) e b)**

Heuristicas propostas:

h1 em que compara-se a diferença entre X atual e o X final, e a diferença entre o Y atual e o Y final. Soma-se o valor que é a distância do fim, e assim optando por um caminho mais curto. Este caminho segue a ordem das operações definidas no caso de igualdade de custo até ao final.





h2 em que se verifica a distância de Y atual ao Y final, e assim opta por se mover para a Esquerda OU Direita e só depois em caso de problema se mover para cima ou para baixo.

