Agente racional para o Mundo do Wumpus

Sistemas Inteligentes

Prof. Heloina Alves Arnaldo

Rodrigo Carlos Carvalho Lima Barbosa Leal

Ricardo Bruno Ferreira da Silva

Representação do cenário

- A matriz bidimensional é a estrutura usada para armazenamento do cenário.

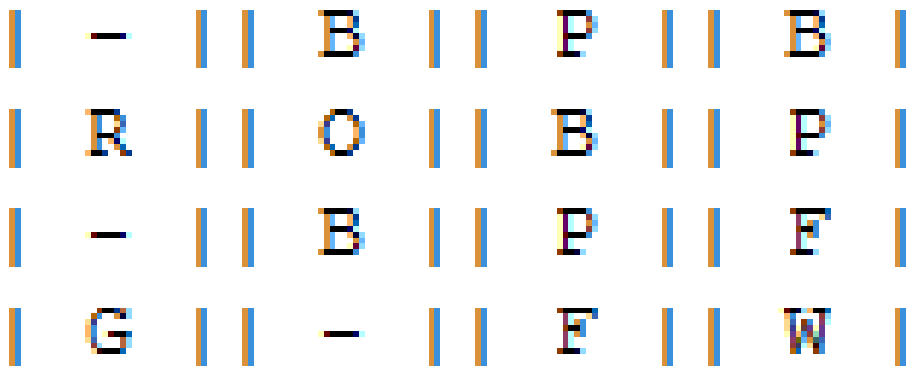
16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Matriz 4x4

- Sendo que cada posição da matriz tem capacidade de armazenar um caractere representativo de um personagem ou percepção.

Representação do cenário

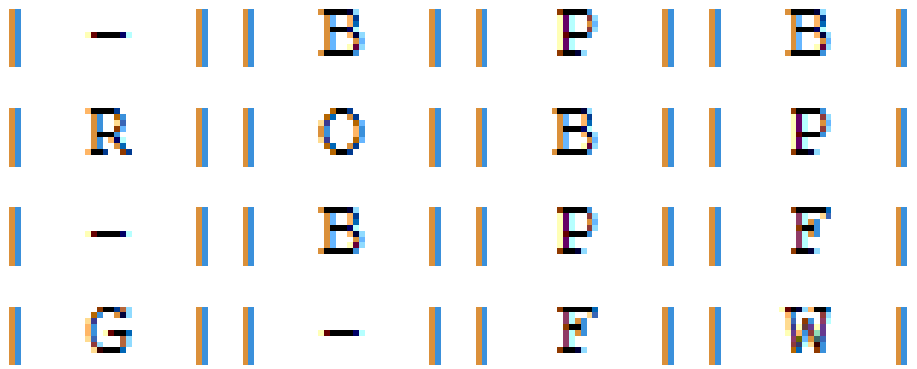
- Caracteres:
  - ❑ G : guerreiro.
  - ❑ W : wumpus.
  - ❑ F : fedor.
  - ❑ B : brisa.
  - ❑ P : poço.
  - ❑ R : brilho.
  - ❑ O : ouro.
  - ❑ - : ausência de percepções.



Representação do cenário

- Ordem de precedência de apresentação dos caracteres de percepção:

1. F;
2. B;
3. R;
4. - .



Isto é possível no algoritmo, dado que, estruturas auxiliares são utilizadas para solucionar o problema comentado.

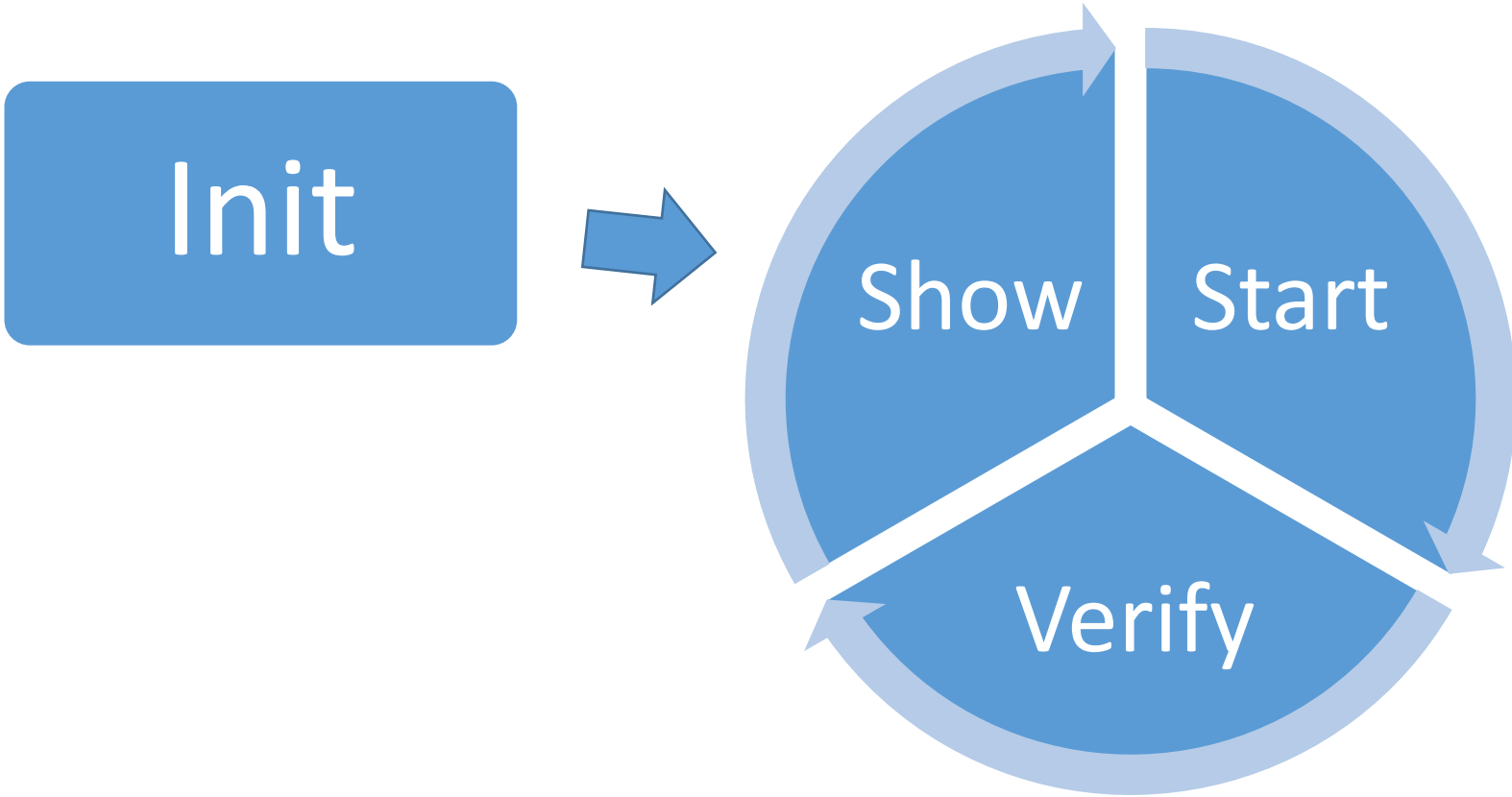
Representação do cenário

static char *brisas[][] = new char[4][4];*
// Esta matriz fica responsável por armazenar as brisas.

static char *fedores[][] = new char[4][4];*
// Esta matriz fica responsável por armazenar os fedores.

static char *brilhos[][] = new char[4][4];*
// Esta matriz fica responsável por armazenar os brilhos.

Funções de Execução



Init

// Função init: inicia o ambiente, colocando o Wumpus,
// Agente, brisas, Fedor e Poços nos seus lugares.

// O ambiente é gerado randomicamente, conforme pedido feito na
// descrição do trabalho da disciplina.

```
public static void init (char ambiente[][]){
...
}
```

Show

// Função show: apresente a matriz passada por parâmetro.

```
public static void show (char ambiente[][]){
        for(int i=0; i<4; i++){
                for(int j=0; j<4; j++)
                        System.out.print("| "+ambiente[i][j]+" |");
                System.out.println(" ");
        }
}
```

Start

// Aqui ocorre o raciocínio e movimento do agente.

public static char[][] start(char ambiente[][]){

…

}

- Uso de estruturas de armazenamento para subsidiar a tomada de decisão do agente do mundo de Wumpus.

Estruturas auxiliares

static ArrayList<Position> ***areas_visitadas = new ArrayList<Position>();***
// Aqui são armazenadas as áreas já visitadas pelo agente.

static ArrayList<Position> ***areas_perigosas = new ArrayList<Position>();***
// Aqui são armazenadas áreas perigosas descobertas pelo agente.

static ArrayList<Position> ***areas_seguras = new ArrayList<Position>();***
// Aqui ficam as áreas consideradas seguras para o agente.

static ArrayList<Position> ***areas_possiveis = new ArrayList<Position>();***
// Neste array ficam as possíveis jogadas ou movimentos do agente.

Start

- A função start busca comparar sempre as áreas seguras, com as áreas visitadas e as áreas possíveis de movimentação.

- Se há uma área segura, que ainda não foi visitada, e que é uma área possível de movimentação, então o movimento é realizado.

- Neste momento, o guerreiro fala: ***Estou me movimentando para uma área que sei que é segura.***

Start

- Entretanto, se as áreas seguras todas já foram visitadas, não resta mais nada para o agente a não ser sortear a sua próxima jogada.

- A próxima jogada é sorteada de forma completamente aleatória usando os recursos da classe Random da linguagem Java.

- Nesta ocorrência, o guerreiro fala: ***Neste momento não sei para onde ir! Vou escolher ao acaso. Então me deseje sorte.***

Estruturas auxiliares

static char *imaginacaoWumpus[][] = new char[4][4];*
// Aqui é uma matriz que mostra a imaginação do Agente, o que o
// Agente tem de interpretação do
// cenário sobre a localização do Wumpus está aqui!

static char *imaginacaoPocos[][] = new char[4][4];*
// Aqui é uma matriz que mostra a imaginação do Agente, o que o
// Agente tem de interpretação do
// cenário sobre a localização dos poços está aqui!

Estruturas auxiliares

- **static char *imaginacaoOuro[][] = new char[4][4];***

// Aqui é uma matriz que mostra a imaginação do Agente, o que o
// Agente tem de interpretação do
// cenário sobre a localização do Ouro está aqui!

Estruturas auxiliares

- A imaginacaoOuro se beneficia das percepções do agente para tentar tirar a conclusão de onde está o Ouro.

- No momento de descoberta do Ouro, o guerreiro (ou agente) fala: **Eu sei onde está o Ouro. Vou pegá-lo!**.

Estruturas auxiliares

- A lista imaginacaoWumpus corresponde a uma imaginação simulada virtualmente do agente sobre o Wumpus que é constantemente alimentada conforme caminhada do guerreiro.

- No momento de descoberta do Wumpus, o agente fala: **Eu sei onde está o Wumpus. Vou matá-lo!**

- Então, o guerreiro ataca com a flecha, e é ouvido o grito do Wumpus.

Estruturas auxiliares

- Já a lista imaginacaoPocos fica responsável por tentar identificar onde ficam os poços do cenário, a partir das percepções do agente.

- Além desta identificação, essas áreas ficam sendo consideradas áreas perigosas, desta forma, o agente não pode visita-las, a não ser em casos excepcionais (quando não houver nenhuma possibilidade).

Estruturas auxiliares

- Situações que ocorre a criação de áreas perigosas:

  - ❑ Quando o guerreiro encontra um fedor;
  - ❑ Quando o guerreiro encontra uma brisa.

Nestas situações, o algoritmo cria áreas perigosas, ou melhor dizendo, áreas suspeitas de perigo em que o guerreiro não vai arriscar entrar nestas, a não ser que não exista qualquer outra possibilidade.

Verify

// Esta função fica responsável por atualizar o status (medida de
// desempenho) do jogo.
// Além de verificar se o guerreiro encontrou o ouro (teste de objetivo).

public static boolean verify(char ambiente[][]){
…
}

Dúvidas?

Agente racional para o Mundo do Wumpus

Sistemas Inteligentes

Prof. Heloina Alves Arnaldo

Rodrigo Carlos Carvalho Lima Barbosa Leal

Ricardo Bruno Ferreira da Silva