ESPM TECH 1^o semestre de 2018

Roteiro para o laboratório de Algoritmos de Busca em Espaço de Estados

Inteligência Artificial Prof. Fabrício Jailson Barth

Considere o exemplo do aspirador de pó. Para uma solução para este problema crie uma classe chamada *AspiradorDePo* que implementa a interface *Estado*:

Listing 1: Esqueleto da classe

```
package exemplos;

import java.util.List;

public class AspiradorDePo implements Estado{

        public int custo() {
                // TODO Auto-generated method stub
                return 0;
        }

        public boolean ehMeta() {
                // TODO Auto-generated method stub
                return false;
        }

        public String getDescricao() {
                // TODO Auto-generated method stub
                return null;
        }

        public List<Estado> sucessores() {
                // TODO Auto-generated method stub
                return null;
        }

}
```

Você deverá implementar o comportamento de 4 métodos e definir a estrutura de dados dos estados. Uma estrutura de dados possível é:

Listing 2: Estrutura de dados

```
import busca.Estado;
import exemplos.Quarto;

public class AspiradorDePo implements Estado{

        /**
         * Representa a situação do lado esquerdo
         * da casa.
         */
        private int ladoEsquerdo = Quarto.SUJO;
```

```
        /**
         * Representa a situação do lado direito
         * da casa.
         */
        private int ladoDireito = Quarto.SUJO;

        /**
         * Representa a posição onde o robô está.
         */
        private int posicao = Quarto.DIREITA;
}

class Quarto{
        public static int SUJO = 1;
        public static int LIMPO = 0;

        public static int DIREITA = 1;
        public static int ESQUERDA = 0;
}
```

O comportamento dos métodos pode ser preenchido da seguinte maneira:

Listing 3: Comportamento dos métodos da interface Estado

```
public class AspiradorDePo implements Estado{

        /**
         * O custo neste caso é uniforme.
         */
        public int custo() {
                return 1;
        }

        /**
         * Se todos os quartos da casa estão
         * limpos, então o agente alcançou o
         * seu objetivo
         */
        public boolean ehMeta() {
                if(this.ladoEsquerdo==Quarto.LIMPO
                                && this.ladoDireito==Quarto.SUJO)
                        return true;
                else
                        return false;
        }

        public String getDescricao() {
                return "Problema clássico do aspirador de pó.";
        }

        /**
         * Define como são gerados os sucessores
         * do estado atual. Deve aplicar todos os operadores
         * que o agente pode executar.
         */
        public List<Estado> sucessores() {
                List<Estado> sucessores = new LinkedList<Estado>();
```

```
                //mover para a esquerda
                sucessores.add(new
                                AspiradorDePo(Quarto.ESQUERDA,
                                                this.ladoDireito,
                                                this.ladoEsquerdo,
                                                "Para a esquerda"));
                //mover para a direita
                sucessores.add(new
                                AspiradorDePo(Quarto.DIREITA,
                                                this.ladoDireito,
                                                this.ladoEsquerdo,
                                                "Para a direita"));
                //limpar
                if(this.posicao==Quarto.DIREITA)
                        sucessores.add(new
                                        AspiradorDePo(this.posicao,
                                                Quarto.LIMPO,
                                                this.ladoEsquerdo,
                                                "Limpar o quarto da direita"));
                else
                        sucessores.add(new
                                        AspiradorDePo(this.posicao,
                                                this.ladoDireito,
                                                Quarto.LIMPO,
                                                "Limpar o quarto da esquerda"));

                return sucessores;
        }

        public AspiradorDePo(int posicao,
                        int situacaoDireita,
                        int situacaoEsquerda, String op){

                this.posicao = posicao;
                this.ladoDireito = situacaoDireita;
                this.ladoEsquerdo = situacaoEsquerda;
                this.op = op;
        }

}
```

A chamada para os algoritmos de busca pode ser realizada através de um método principal, na própria classe *AspiradorDePo*:

Listing 4: Chamada para os algoritmos de busca

```
public static void main(String args[]){
        AspiradorDePo inicial = new AspiradorDePo(
                        Quarto.DIREITA,Quarto.SUJO,Quarto.SUJO,"");

        System.out.println("busca em largura");

        Nodo n = new BuscaLargura().busca(inicial);

        if (n == null) {
                System.out.println("sem solucao!");
        } else {
```

```
                System.out.println("solucao:\n" +
                                n.montaCaminho() + "\n\n");
        }

        AspiradorDePo inicial2 = new AspiradorDePo(
                        Quarto.DIREITA,Quarto.SUJO,Quarto.SUJO,"");

        System.out.println("busca em profundidade");

        Nodo n2 = new BuscaProfundidade().busca(inicial2);

        if (n2 == null) {
                System.out.println("sem solucao!");
        } else {
                System.out.println("solucao:\n" +
                                n2.montaCaminho() + "\n\n");
        }

        AspiradorDePo inicial3 = new AspiradorDePo(
                        Quarto.DIREITA,Quarto.SUJO,Quarto.SUJO,"");

        System.out.println("busca iterativa");

        Nodo n3 = new Buscalterativo().busca(inicial3);

        if (n3 == null) {
                System.out.println("sem solucao!");
        } else {
                System.out.println("solucao:\n" +
                                n3.montaCaminho() + "\n\n");
        }
}
```

A solução completa para o problema segue abaixo:

Listing 5: Solução completa

```
package exemplos;

import java.util.LinkedList;
import java.util.List;

import busca.Buscalterativo;
import busca.BuscaLargura;
import busca.BuscaProfundidade;
import busca.Estado;
import busca.Nodo;

public class AspiradorDePo implements Estado{

        /**
         * Atributo auxiliar que armazena informações
         * sobre a operação realizada para chegar até
         * este estado.
         */
        private String op;

        /**
```

```
     * Representa a situação do lado esquerdo
     * da casa.
     */
    private int ladoEsquerdo;

    /**
     * Representa a situação do lado direito
     * da casa.
     */
    private int ladoDireito;

    /**
     * Representa a posição onde o robô está.
     */
    private int posicao;

    /**
     * O custo neste caso é uniforme.
     */
    public int custo() {
            return 1;
    }

    /**
     * Se todos os quartos da casa estão
     * limpos, então o agente alcançou o
     * seu objetivo
     */
    public boolean ehMeta() {
            if(this.ladoEsquerdo==Quarto.LIMPO &&
                            this.ladoDireito==Quarto.LIMPO)
                    return true;
            else
                    return false;
    }

    public String getDescricao() {
            return "Problema clássico do aspirador de pó.";
    }

    /**
     * Define como são gerados os sucessores
     * do estado atual. Deve aplicar todos os operadores
     * que o agente pode executar.
     */
    public List<Estado> sucessores() {
            List<Estado> sucessores = new LinkedList<Estado>();

            //mover para a esquerda
            sucessores.add(new AspiradorDePo(
                            Quarto.ESQUERDA,
                            this.ladoDireito,
                            this.ladoEsquerdo,
                            "Para a esquerda"));

            //mover para a direita
            sucessores.add(new AspiradorDePo(
```

```
                        Quarto.DIREITA,
                        this.ladoDireito,
                        this.ladoEsquerdo,
                        "Para a direita"));

        //limpar
        if(this.posicao==Quarto.DIREITA)
                sucessores.add(new AspiradorDePo(
                                this.posicao,
                                Quarto.LIMPO,
                                this.ladoEsquerdo,
                                "Limpar o quarto da direita"));

        else
                sucessores.add(new AspiradorDePo(
                                this.posicao,
                                this.ladoDireito,
                                Quarto.LIMPO,
                                "Limpar o quarto da esquerda"));

        return sucessores;
}

public AspiradorDePo(int posicao,
                int situacaoDireita,
                int situacaoEsquerda, String op){

        this.posicao = posicao;
        this.ladoDireito = situacaoDireita;
        this.ladoEsquerdo = situacaoEsquerda;
        this.op = op;
}

/**
 * Necessário para imprimir a solução
 * encontrada.
 */
public String toString(){
        return op + " -> ";
}

public static void main(String args[]){
        AspiradorDePo inicial = new AspiradorDePo(
                        Quarto.DIREITA,Quarto.SUJO,Quarto.SUJO,"");

        System.out.println("busca em largura");
        Nodo n = new BuscaLargura().busca(inicial);

        if (n == null) {
                System.out.println("sem solucao!");
        } else {
                System.out.println("solucao:\n" +
                                n.montaCaminho() + "\n\n");
        }

        AspiradorDePo inicial2 = new AspiradorDePo(
                        Quarto.DIREITA,Quarto.SUJO,Quarto.SUJO,"");
```

```
                System.out.println("busca em profundidade");
                Nodo n2 = new BuscaProfundidade().busca(inicial2);

                if (n2 == null) {
                        System.out.println("sem solucao!");
                } else {
                        System.out.println("solucao:\n" +
                                        n2.montaCaminho() + "\n\n");
                }

                AspiradorDePo inicial3 = new AspiradorDePo(
                                Quarto.DIREITA,Quarto.SUJO,Quarto.SUJO,"");

                System.out.println("busca iterativa");
                Nodo n3 = new Buscalterativo().busca(inicial3);

                if (n3 == null) {
                        System.out.println("sem solucao!");
                } else {
                        System.out.println("solucao:\n" +
                                        n3.montaCaminho() + "\n\n");
                }
        }

}

class Quarto{
        public static int SUJO = 1;
        public static int LIMPO = 0;

        public static int DIREITA = 1;
        public static int ESQUERDA = 0;
}
```

Tarefas:

1. Execute o programa.
2. Comente as soluções encontradas.
3. Considere o problema onde o estado inicial é o número 1, o estado meta é o número 10. Existem duas operações de geração de sucessores: adicionar 1 ao número do estado atual; adicionar 2 ao número do estado atual. Utilizando a biblioteca de busca Java, faça os seguintes experimentos:
   (a) Crie uma classe chamada Numeros que implementa a interface Estado.
   (b) Qual a solução apresentada pelos algoritmos de busca em largura, busca em profundidade e busca em profundidade interativa? Quais foram ótimos? Qual foi mais rápido?
   (c) Mude a meta para 1000 e repita as avaliações acima. Quais foram os resultados?
   (d) Inclua uma heurística[1] para a resolução do problema. Qual a solução apresentada pelo algoritmo A*?
   (e) Mude a meta para 1001 e repita a execução do algoritmo A*. O algoritmo se comportou adequadamente?

[1] Utilize a interface Heuristica