Universidade Federal de Ouro Preto
Inteligência Artificial
Teste - Busca em Espaço de Estados

Prof. Rodrigo Silva

1. Quais algoritmos de busca em espaço de estados você utilizaria para encontrar o caminho de menor custo entre um estado inicial e um estado final (ou meta)? Compare os algoritmos selecionados em termos de custo computacional (tempo de execução e espaço de memória) e apresente vantagens e desvantagens na utilização de cada um dos métodos.

2. Selecione a opção correta para cada célula da tabela. $h(n)$ é o valor da função heurística do nó n. $c(S, n)$ é o custo do caminho de um nó S até o nó n.

Estratégia	Seleção da fronteira	Caminho Encontrado	Custo em Espaço
Busca em Largura			
Busca em Profundidade			
Guloso			
Menor Caminho Primeiro			
A^*			
Branch and Bound			

   (a) Menor $h(n)$
   (b) Menor $c(S, n)$
   (c) Menor $h(n) + c(S, n)$
   (d) Primeiro caminho adicionado
   (e) Último caminho adicionado
   (f) Menor número de arcos
   (g) Indefinido
   (h) Menor custo
   (i) Linear
   (j) Exponencial

3. Para o itens abaixo, considere o código a seguir que apresenta a implementação de um agente que faz uma busca encontrar a saída de um labirinto.

   (a) Qual algoritmo de busca este agente implementa?
   (b) Modifique o método `act` de forma que o agente implemente poda de ciclos.
   (c) Modifique o método `act` de forma que o agente implemente poda de múltiplos caminhos.
   (d) Modifique o método `act` de forma que o agente implemente um outro método de busca sem informação.
   (e) Modifique o método `act` de forma que o agente implemente o método de busca A*.

```python
class MazeAgentDFS():

    def __init__(self,env):
        self.env = env
        self.percepts = env.initial_percepts()
        self.F = [[self.percepts['position']]]

    def act(self):

        while self.F:
            path = self.F.pop(-1)

            self.percepts = self.env.change_state({'path':path.copy()})

            if self.percepts['goal']:
                break
            else:
                for n in self.percepts['available_neighbors']:
                    self.F.insert(-1,path + [n])

        self.env.draw_best(path)
```

Figure 1: DFS agent