Lista de Exercícios – Busca em Espaço de Estados

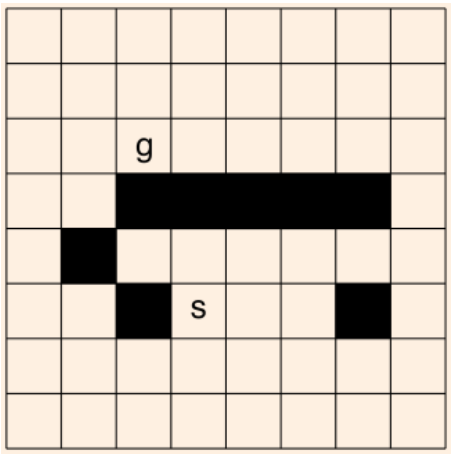
BCC740 – Inteligência Artificial
Prof. Rodrigo Silva

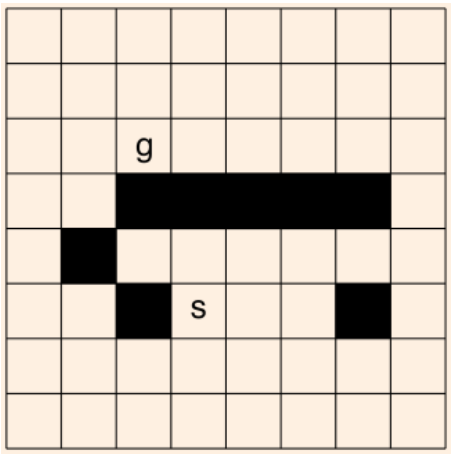
Leitura Recomendada

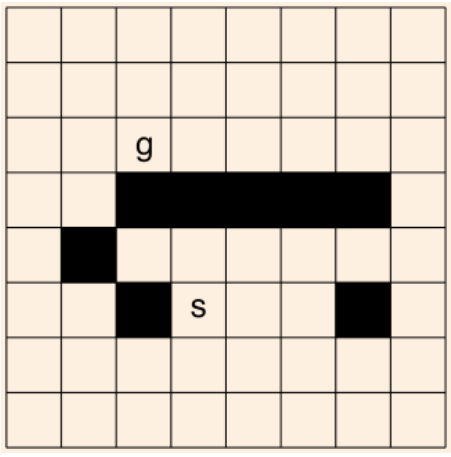
- Capítulo 3 (*Problem Solving as Search*) do livro **Poole & Mackworth, Artificial Intelligence: Foundations of Computational Agents (3e)**, disponível em `https://artint.info/`.

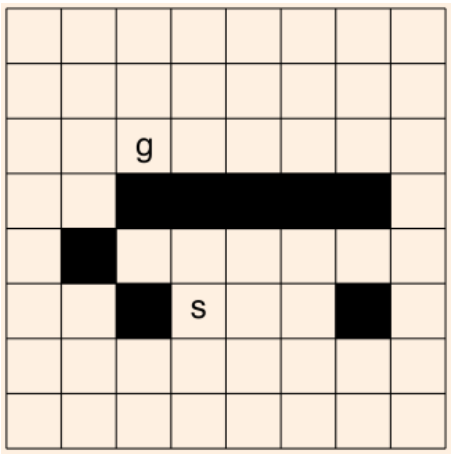
1 Questões Teóricas

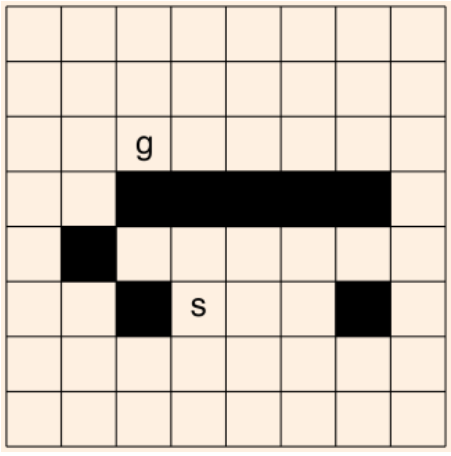
1. (Cap. 1) O que é um **agente**? Explique como ele interage com o ambiente.
2. (Cap. 1) Quando dizemos que um agente é **inteligente**? Quais propriedades o caracterizam como tal?
3. (Seção 3.1) O que significa **busca** no contexto dos métodos de resolução de problemas apresentados no livro?
4. (Seção 3.2) Quais são as **premissas** de um problema de busca em espaços de estados?
5. (Seção 3.2) Quais são os **componentes** formais de um problema de busca em espaços de estados?
6. (Seção 3.3) Qual a relação entre **espaços de estados** e **grafos**?
7. (Seção 3.3.1) Quais são os **componentes e objetivos** de um problema de busca em grafo?
8. (Seção 3.4) Apresente o **algoritmo genérico de busca** e descreva suas principais estruturas de dados.
9. (Seção 3.5.1) Mostre um exemplo de execução do **algoritmo de busca em profundidade (DFS)**. Apresente o estado da fronteira a cada iteração.
10. (Seção 3.5.2) Mostre um exemplo de execução do **algoritmo de busca em largura (BFS)**. Apresente o estado da fronteira a cada iteração.
11. (Seção 3.5) Apresente uma discussão sobre os custo de**tempo e espaço** dos algoritmos **DFS** e **BFS**.
12. Considere o problema de encontrar um caminho no labirinto abaixo. O objetivo é ir da posição **s** até a posição **g**. O agente pode se mover nas quatro direções (cima, baixo, esquerda, direita).
    - 12.a) Numere os nós expandidos (visitados) por um agente que implementa **busca em profundidade**. A ordem das ações é: cima, esquerda, direita e baixo. (Assuma poda de ciclos.)
    - 12.b) Numere os nós expandidos por um agente que implementa a **busca de menor custo primeiro**.
    - 12.c) Escreva em cada nó o valor da **heurística de distância de Manhattan** (distância em unidades horizontais + verticais até o objetivo).
    - 12.d) Numere os nós expandidos por um agente que implementa a **busca gulosa** utilizando a heurística acima. (Assuma poda de ciclos.)
    - 12.e) Numere os nós expandidos por um agente que implementa o algoritmo A, considerando a distância de Manhattan como custo e heurística.




g
s


g
s


g
s


g
s

2 Trabalho Prático – Missionários e Canibais

Neste trabalho, o estudante deverá implementar um agente que resolva o **problema dos missionários e canibais**, conforme discutido em aula.

Descrição do Problema

Três missionários e três canibais estão em uma margem do rio e precisam atravessar usando um barco que comporta no máximo duas pessoas. Em nenhum momento pode haver mais canibais do que missionários em uma margem (exceto quando não houver missionários nela).

O problema deve ser resolvido como um **problema de busca**, utilizando as representações:

$$(M_E, C_E, B)$$

onde M_E é o número de missionários à esquerda, C_E é o número de canibais à esquerda, e $B \in \{E, D\}$ indica a posição do barco.

O que deve ser implementado

1. Crie uma classe `MissionariesAndCannibals`, com:
   - Representação do estado inicial e estado meta;
   - Método `successors(state)` que gera todos os estados válidos (sem violar as restrições);
   - Método `is_goal(state)` que testa se o estado é uma solução;
   - Custo unitário por travessia.
2. Implemente versões de **BFS**, **DFS** e **UCS** (utilizando o algoritmo genérico de busca apresentado em aula).
3. Mostre a sequência de ações e o número de nós expandidos para cada método.

Variações

Explore versões modificadas do problema e observe como o espaço de estados muda:

- Mude a **capacidade do barco** (ex.: 3 lugares);
- Altere o número de missionários e canibais (ex.: 4 de cada lado);
- Atribua **custos diferentes** para certas travessias (ex.: atravessar com dois canibais custa mais);
- Adicione **restrições assimétricas** (ex.: o barco deve sempre retornar com alguém);
- Faça comparações entre o desempenho dos algoritmos para cada variação.

Entrega

- `missionaries.py` com a implementação do problema;
- `agents.py` com as funções de busca adaptadas;
- Um arquivo `main.py` que execute os testes e exiba a sequência de ações e o número de nós expandidos.