Universidade Federal de Ouro Preto
BCC 325 - Inteligência Artificial
Busca em Espaço de Estados

Prof. Rodrigo Silva

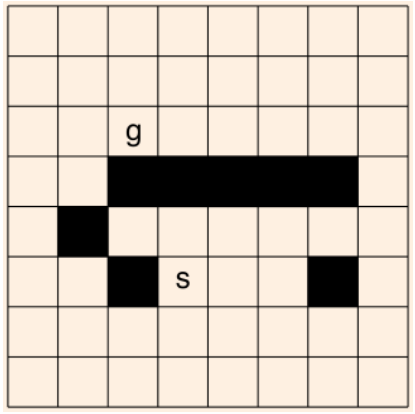
1 Leitura

- Capítulo 3 do Livro *Artificial Intelligence: Foundations of Computational Agents, 2nd Edition* disponível em *https://artint.info/*

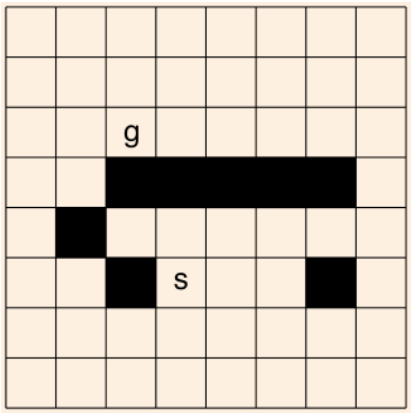
2 Questões

1. (Seção 3.1) O que significa busca no contexto dos métodos da leitura proposta?
2. (Seção 3.2) Quais são as premissas de um problem de busca em espaços de estados?
3. (Seção 3.2) Quais são os componentes de um problema de busca em espaços de estados?
4. (Seção 3.3) Qual a relação entre espaços de estados e grafos?
5. (Seção 3.3.1) Quais os componente e os objetivos de um problema de busca em grafos?
6. (Seção 3.4) Apresente o algoritmo genérico de busca?
7. (Seção 3.5.1) Apresente um exemplo de execução do algoritmo de busca em profundidade. (Apresente o estado da fronteira a cada interação.)
8. (Seção 3.5.2) Apresente um exemplo de execução do algoritmo de busca em largura.(Apresente o estado da fronteira a cada interação.)
9. 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 horizontalmente e verticalmente.

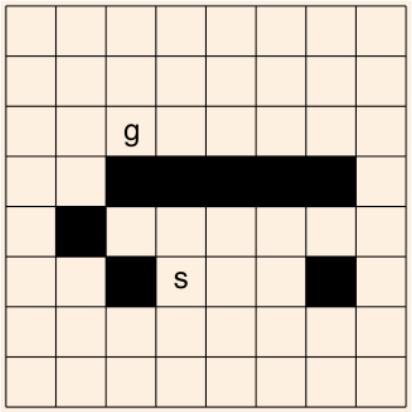
   (a) No labirinto abaixo, numere os nós expandidos (visitados) por um agente que implementa o algoritmo de busca e profundidade. A ordem das ações é para cima, para a esquerda, para a direita, e para baixo. Assuma poda de ciclos.



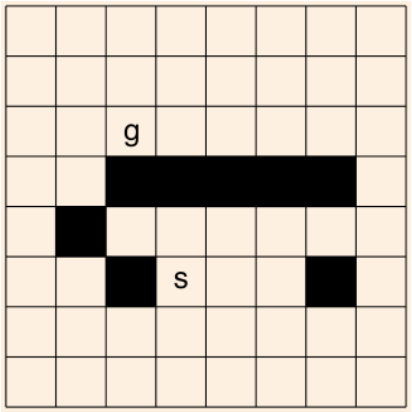
(b) No labirinto abaixo, numere os nós expandidos (visitados) por um agente que implementa um algoritmo de busca de Menor custo primeiro.



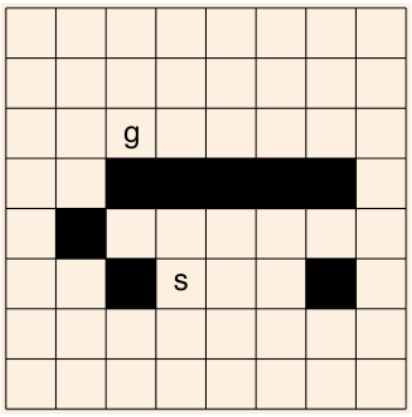
(c) No labirinto abaixo, escreva em cada nó o valor da heurística do nó, considerando a distância de Manhattan. Considere que cada quadrado tem lado 1 u.m.



(d) No labirinto abaixo, numere os nós expandidos (visitados) por um agente que implementa um algoritmo guloso pela heurística calculada acima. Assuma poda de ciclos.



(e) (1pt) No labirinto abaixo, numere os nós expandidos (visitados) por um agente que implementa o algoritmo A^* considerando a distância de Manhattan como custo e heurística.



3 Trabalho Prático

Neste trabalho, você irá modificar os arquivos `agents.py` e `environment.py` para resolver o problema de encontrar um caminho em um labirinto bidimensional, representado como uma matriz.

Descrição do Problema

O labirinto será representado por uma matriz $m \times n$, onde cada célula pode conter:

- `0`: espaço livre;
- `1`: parede (bloqueio);
- `s`: posição inicial;
- `g`: posição objetivo.

O agente pode se mover para cima, baixo, esquerda ou direita, desde que o movimento não ultrapasse os limites da matriz nem atravesse uma parede.

O que deve ser feito

1. Modifique o arquivo `environment.py`, adicionando uma nova classe chamada `MazeEnvironment`, que:
   - Receba uma matriz como entrada;
   - Identifique automaticamente a posição inicial (`s`) e a posição objetivo (`g`);
   - Implemente o método `get_neighbors(state)` que retorna apenas os vizinhos válidos de uma célula.
2. Modifique as classes `BFS` e `DFS` do arquivo `agents.py` para funcionarem com posições no formato `(linha, coluna)`.
3. Teste sua implementação utilizando o arquivo `maze_simulation.py`, que será fornecido. Este arquivo:

- Cria um labirinto de exemplo;
- Executa o algoritmo de busca (BFS ou DFS);
- Imprime o caminho encontrado;
- Mostra o labirinto original com o caminho marcado com o símbolo `*`.

Dicas

- Use uma representação de posições como tuplas: `(linha, coluna)`;
- Faça uma cópia da matriz original para marcar o caminho encontrado;
- A função `get_neighbors` deve verificar se a nova posição está dentro da matriz e se não é uma parede;

Entrega

No seu repositório do GitHub você deve criar um pasta "TrabalhoPraticoBusca" que deve conter:

- `agents.py` modificado;
- `environment.py` modificado;
- O arquivo `maze_simulation.py` fornecido para testes.

O professor dará instruções de como enviar o link do repositório durante a aula.