PUC Minas

Trabalho em grupo 1 - Valor 10 pontos

INFORMAÇÕES DOCENTE						
CURSO: ENGENHARIA DE SOFTWARE	**DISCIPLINA:** FUNDAMENTOS DE PROJETO E ANÁLISE DE ALGORITMOS	**TURNO**	MANHÃ	TARDE	NOITE	**PERÍODO/SALA:** 5º
					x	
PROFESSOR (A): João Paulo Carneiro Aramuni						

Enunciado do projeto: **PathFinder - Resolvendo o Labirinto 2D com o Algoritmo A***

Contexto:

- Seu grupo foi contratado para desenvolver um algoritmo que ajudará um robô de resgate a encontrar o menor caminho dentro de um labirinto.
- O robô está posicionado em um ponto inicial *A* e precisa chegar ao ponto final *B*, movendo-se dentro do labirinto sem colidir com obstáculos.
- O algoritmo que será usado para resolver esse problema é o A*, que combina o custo do caminho já percorrido e uma estimativa (heurística) da distância até o destino para encontrar a solução de forma eficiente.

Objetivo:

- Implementar o Algoritmo A* para encontrar o menor caminho em um labirinto 2D entre dois pontos, evitando obstáculos e considerando os custos dos movimentos.

Regras do Labirinto:

- 1. O labirinto é representado por uma matriz 2D, onde:
  - 0: Células livres (onde o robô pode se mover).
  - 1: Obstáculos (onde o robô não pode passar).
  - S: Ponto inicial (start).
  - E: Ponto final (end).

  Exemplo de labirinto:

```
S 0 1 0 0
0 0 1 0 1
0 1 0 0 0
1 0 0 E 1
```

- 2. O robô pode se mover para as células adjacentes (cima, baixo, esquerda e direita), desde que a célula não seja um obstáculo ou esteja fora dos limites do labirinto.

- 3. O custo de cada movimento é sempre 1.

Regras do trabalho:

- Grupos de até 3 pessoas.
- Todos commitam no código.

Tarefas:

1. Leitura do labirinto:
   - Receber como entrada uma matriz que representa o labirinto.

2. Definição da heurística:
   - Implementar uma função heurística que estima a distância entre o ponto atual e o ponto final. Use a distância de Manhattan:

$$h(n) = |\ x \text{ atual} - x \text{ final}\ | + |\ y \text{ atual} - y \text{ final}\ |$$

3. Implementação do Algoritmo A*:
   - Desenvolver o algoritmo para explorar o labirinto e encontrar o menor caminho entre S e E.

4. Exibição do resultado:
   - Mostrar o caminho encontrado no formato de uma lista de coordenadas.
   - Destacar o caminho no labirinto.

5. Documentação no README.md:
   - Crie um arquivo README.md contendo a descrição do projeto, uma introdução sobre o problema resolvido (labirinto 2D com Algoritmo A*), as instruções necessárias para configurar e executar o projeto, e uma explicação clara sobre o funcionamento do Algoritmo A* implementado, destacando como ele combina o custo do caminho percorrido e a heurística da distância de Manhattan para encontrar o menor caminho no labirinto.
   - Certifique-se de incluir exemplos de entrada e saída para ilustrar o funcionamento do projeto.

Requisitos de implementação:

- A entrada deve ser uma matriz 2D definida pelo usuário ou gerada automaticamente.
- O programa deve validar se S e E existem no labirinto antes de executar o algoritmo.
- O algoritmo deve parar e retornar "Sem solução" caso não haja caminho possível entre S e E.

Exemplo de Entrada e Saída:

Entrada:
Labirinto:

```
S 0 1 0 0
0 0 1 0 1
1 0 1 0 0
1 0 0 E 1
```

Saída:
Menor caminho (em coordenadas):
[s(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), e(3, 3)]

Labirinto com o caminho destacado:

```
S 0 1 0 0
* * 1 0 1
1 * 1 0 0
1 * * E 1
```

Ponto extra (Opcional):

- Permitir que o robô também se mova nas diagonais (com custo √2).
- Implementar uma interface gráfica simples para visualizar o robô explorando o labirinto em tempo real.
- Adicionar pesos diferentes às células livres (ex.: custo de atravessar terrenos difíceis).

Critérios de avaliação:

- Correção do algoritmo (deve encontrar o menor caminho corretamente).
- Clareza e organização do código.
- Comentários explicativos e boas práticas de programação.
- Implementação de ideias extras ou criatividade no problema.

Recursos adicionais:

- Caso queiram testar o algoritmo com labirintos maiores ou mais complexos, vocês podem gerar labirintos automaticamente usando uma biblioteca como a *numpy* ou mesmo criar entradas fixas para verificar diferentes casos.

Entrega:

- O projeto deverá ser enviado por meio de um repositório no GitHub, com o link postado no sistema CANVAS. Certifique-se de que o repositório esteja público

ou acessível (antes de realizar a entrega, faça um teste em uma aba anônima do navegador).

- Preferencialmente, todos os integrantes do grupo entregam o link do repositório do trabalho no CANVAS. Isto é útil para que o registro de entrega fique salvo em cada usuário.
- Exemplo de link a ser entregue no CANVAS:
  - https://github.com/exemploaluno/trabalho_em_grupo_1_FPAA

Critérios de avaliação:

1. Implementação do algoritmo (50%):
   - O código está correto e eficiente?
   - A lógica para encontrar o melhor caminho, por meio do Algoritmo A*, foi seguida adequadamente?
   - O algoritmo é capaz de lidar corretamente com diferentes entradas, como labirintos complexos, labirintos sem solução e labirintos grandes?
   - O código possui clareza, está organizado e segue boas práticas de programação?

2. Documentação no README.md (50%):
   - O README segue o padrão especificado?
   - A documentação fornece instruções claras para configurar e executar o projeto?
   - A explicação do Algoritmo A* está detalhada e compreensível, com exemplos de entrada e saída?
   - O README contém informações suficientes para que qualquer pessoa possa entender e utilizar o projeto sem dificuldades?
