

Trabalho em grupo 2 - Valor 10 pontos

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

Enunciado do projeto: **FloodFill** - Colorindo regiões de um terreno com obstáculos

Contexto:

- Sua equipe foi contratada por uma empresa de automação para desenvolver um sistema de mapeamento inteligente para robôs autônomos que precisam identificar e classificar regiões de um terreno previamente desconhecido.
- O terreno é representado como um grid bidimensional, onde cada célula pode ser um espaço livre ou um obstáculo. Além disso, diferentes áreas do terreno podem ser separadas por barreiras, formando regiões desconectadas.
- O sistema deve identificar e "colorir" automaticamente todas as áreas livres conectadas, de forma a facilitar a visualização e o planejamento do robô para executar suas operações.

Objetivo:

- Implementar o Algoritmo Flood Fill para identificar e preencher automaticamente todas as regiões conectadas em um grid 2D, utilizando diferentes cores para cada área.
- O algoritmo deve respeitar os obstáculos presentes no terreno e evitar colidir com eles.
- O programa também deve ser capaz de localizar novas áreas livres automaticamente após preencher uma região, continuando o processo até que todo o terreno esteja mapeado e colorido.

Descrição do problema:

- Você receberá um grid bidimensional  $n \times m$ , onde cada célula pode conter um dos seguintes valores:
  - 0: Terreno navegável (regiões que podem ser preenchidas - branco).
  - 1: Obstáculo (não navegável, deve ser ignorado pelo preenchimento - preto).
  - 2, 3, 4, ...: Cores já preenchidas em outras regiões (vermelho, laranja, amarelo, etc.).

- Além disso, será fornecida uma célula inicial  $(x, y)$ , de onde o preenchimento começará.
- O algoritmo deve:
  - I. Determinar todas as células conectadas à célula inicial que possuem o valor 0.
  - II. Substituir o valor 0 dessas células por um valor de cor específico (ex.: 2 para a primeira região preenchida, 3 para a próxima, e assim por diante).

#### Requisitos do programa:

##### 1. Entrada:

- Dimensões do grid  $n \times m$ .
- O grid em si, com os valores 0, 1, e, opcionalmente, cores já preenchidas.
- Coordenadas iniciais  $(x, y)$  para o preenchimento.

##### 2. Saída:

- O grid atualizado, com a região conectada à célula inicial preenchida pela cor fornecida.

##### 3. Regras:

- O preenchimento deve respeitar os obstáculos (células com valor 1) e não pode ultrapassá-los.
- Regiões já coloridas (valores acima de 1) devem ser mantidas intactas.
- A região conectada é composta apenas por células navegáveis (0) adjacentes ortogonalmente (acima, abaixo, à esquerda e à direita).
- Após o preenchimento de uma região, o programa deve localizar automaticamente a próxima célula navegável (0) e preencher uma nova região com uma cor diferente (incrementando o valor da cor: 2, 3, 4, ...).
- O processo deve continuar até que todas as células navegáveis tenham sido preenchidas.

##### 4. Documentação no README.md:

- Crie um arquivo README.md contendo a descrição do projeto, uma introdução sobre o problema resolvido (identificação e preenchimento de regiões conectadas em um grid 2D com o Algoritmo Flood Fill), as instruções necessárias para configurar e executar o projeto, e uma explicação clara sobre o funcionamento do Algoritmo Flood Fill implementado.
- Destaque como o algoritmo percorre o grid a partir de uma célula inicial, identifica células navegáveis conectadas, e as preenche com uma cor específica, respeitando obstáculos e limites do grid.



**PUC Minas**

- Certifique-se de incluir exemplos claros de entrada e saída para ilustrar o funcionamento do projeto, utilizando grids de diferentes tamanhos e configurações. As representações visuais devem incluir um grid inicial, com terrenos navegáveis e obstáculos, e o grid final, com regiões preenchidas em cores distintas.

Exemplo 1:

Entrada:

Grid inicial:

```
0 0 1 0 0
0 1 1 0 0
0 0 0 1 1
1 1 0 0 0
```

Coordenadas iniciais: (0, 0)

Saída:

Grid preenchido:

```
2 2 1 3 3
2 1 1 3 3
2 2 2 1 1
1 1 4 4 4
```

Exemplo 2:

Entrada:

Grid inicial:

```
0 1 0 0 1
0 1 0 0 1
0 1 1 1 1
0 0 0 1 0
```

Coordenadas iniciais: (0, 2)

Saída:

Grid preenchido:

```
3 1 2 2 1
3 1 2 2 1
3 1 1 1 1
4 4 4 1 5
```

Ponto extra (Opcional):

- Adicionar uma interface gráfica simples que mostre o grid sendo preenchido dinamicamente.
- Gerar automaticamente grids aleatórios com diferentes proporções de terrenos navegáveis e obstáculos.

Visualização com cores:

Exemplo 1:

Entrada:

Grid inicial:

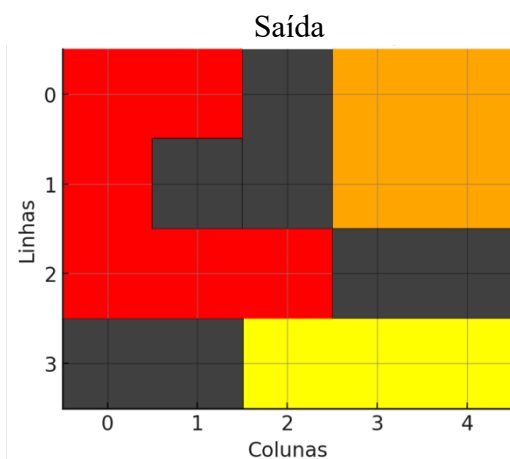
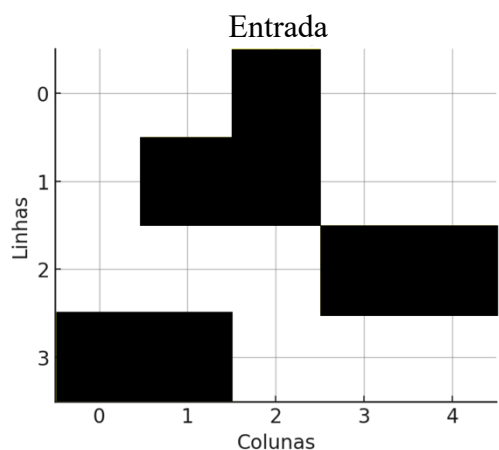
```
0 0 1 0 0
0 1 1 0 0
0 0 0 1 1
1 1 0 0 0
```

Coordenadas iniciais: (0, 0)

Saída:

Grid preenchido:

```
2 2 1 3 3
2 1 1 3 3
2 2 2 1 1
1 1 4 4 4
```



Legenda:

- 0 – Branco (Terreno navegável)
- 1 – Preto (Obstáculo)
- 2 – Vermelho
- 3 – Laranja
- 4 – Amarelo

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\\_2\\_FPAA](https://github.com/exemploaluno/trabalho_em_grupo_2_FPAA)

Critérios de avaliação:

1. Corretude: O algoritmo deve preencher as regiões corretamente, respeitando os obstáculos e delimitadores.
2. Eficiência: Soluções recursivas e iterativas serão avaliadas quanto à eficiência em grids grandes.
3. Visualização - A solução deve apresentar o grid antes e depois do preenchimento, em duas versões:
  - Versão de terminal: No formato impresso de matriz (utilizando números, 0, 1, 2, 3, 4, etc. para as linhas e colunas).
  - Versão gráfica: Colorida, conforme mostrado na *Visualização com cores* do Exemplo 1.

Distribuição de pontos:

1. Implementação do Algoritmo (50%):
  - O código está correto e eficiente?
  - A lógica para identificar e preencher regiões conectadas, por meio do Algoritmo Flood Fill, foi seguida adequadamente?
  - O algoritmo é capaz de lidar corretamente com diferentes entradas, como grids grandes, grids com múltiplos obstáculos e grids sem células navegáveis?
  - 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 Flood Fill está detalhada e compreensível, destacando como o algoritmo identifica regiões conectadas e as preenche com cores distintas, respeitando os obstáculos?
  - O README contém exemplos claros de entrada e saída para ilustrar o funcionamento do projeto?
  - As informações são suficientes para que qualquer pessoa possa entender e utilizar o projeto sem dificuldades?