# Simulação de Propagação de Incêndios

Michel Pires da Silva – CEFET-MG March 27, 2025

**Observações:**

1. O trabalho deve ser realizado individualmente por cada aluno.
2. A entrega deve ser feita até a data estipulada neste documento. Para ser considerada completa, a entrega deve incluir:
   - Documentação abrangente que descreva detalhadamente o problema e a solução proposta. Devem ser incluídos o método de compilação e as referências utilizadas. Todo trabalho que envolver o uso de estruturas de dados deve se basear em fontes científicas que expliquem detalhadamente a estrutura abordada.
   - Apresentação das estruturas de dados utilizadas, com a devida justificativa para suas escolhas.
   - Resultados de desempenho e análise da execução do programa.
   - Conclusões finais sobre o desenvolvimento da solução e possíveis melhorias.
3. O trabalho deve ser obrigatoriamente executável em sistemas operacionais Linux. É responsabilidade do aluno garantir que o programa compile e execute corretamente, corrigindo eventuais erros. A compilação deve seguir os padrões estabelecidos na disciplina, utilizando o arquivo Makefile disponível no Moodle.
4. A avaliação será baseada na documentação e na solução entregues. Portanto, se o programa não for executado corretamente em Linux, essa falha será desconsiderada na avaliação. Além disso, não será permitida a reentrega ou a correção do trabalho após a submissão e avaliação.
5. As linguagens de programação aceitas para a implementação são C e C++.

---

Neste trabalho, cada aluno será responsável pela implementação de um simulador de propagação de incêndio em uma floresta, utilizando a linguagem de programação C ou C++. O objetivo principal consiste em modelar a propagação do fogo em uma matriz, onde cada célula representa um estado da simulação, o qual pode evoluir ao longo da execução, conforme as interações de propagação de incêndio. A implementação deverá permitir a utilização de diferentes estratégias computacionais para a propagação do fogo, proporcionando uma análise comparativa do desempenho e da eficiência das soluções adotadas.

A estratégia de propagação e o número de interações a serem realizadas durante a simulação deverão ser definidos como variáveis globais no sistema, configuráveis por meio de um arquivo de cabeçalho (.h ou .hpp), denominado *config.h* ou *config.hpp*, a depender do paradigma de programação adotado. Demais parâmetros relevantes para a configuração da execução também deverão ser especificados neste arquivo.

Além da simulação da propagação do incêndio, o modelo deverá incluir a funcionalidade de identificação de rotas de fuga para um animal presente na floresta, sempre que possível. O animal poderá se deslocar em qualquer direção ortogonal na matriz, buscando alcançar posições seguras ou fontes de água (representadas pelo valor 4). Ao detectar uma célula com água, a posição do animal será atualizada para um estado seguro (valor 0), e todas as células ortogonalmente adjacentes a essa posição serão convertidas para o estado 1, indicando a presença de árvores saudáveis.

A movimentação do animal possui prioridade sobre a propagação do incêndio, ou seja, ele executa sua ação antes da atualização do fogo na matriz. Após a propagação do incêndio, verifica-se se a posição do animal foi atingida pelo fogo. Caso isso ocorra, o animal receberá uma nova oportunidade de movimentação para tentar escapar. Além disso, o deslocamento do animal na matriz deverá ser monitorado, registrando-se o número total de passos percorridos, a quantidade de vezes em que encontrou água e, caso fique impossibilitado de prosseguir devido ao avanço do fogo, o número da iteração em que isso ocorreu.

Este trabalho tem como objetivo não apenas a implementação técnica do simulador, mas também a reflexão crítica sobre o impacto das escolhas algorítmicas no desempenho da simulação, considerando diferentes estratégias de propagação e movimentação.

# 1 Especificação do Problema

A área de floresta será representada por uma matriz de dimensões $N \times M$, na qual cada célula corresponderá a um quadrante dessa área. Para essa atividade, considere valores mínimos de tamanho a configuração $100$x$100$. Nesse modelo representativo, cada quadrante da matriz poderá conter um dos seguintes valores, representando o estado do item simulado:

- 0 $\rightarrow$ Área vazia (não queima)
- 1 $\rightarrow$ Árvore saudável
- 2 $\rightarrow$ Árvore em chamas
- 3 $\rightarrow$ Árvore queimada (não propaga mais fogo, árvore totalmente queimada)
- 4 $\rightarrow$ Presença de água

A propagação do incêndio segue as regras descritas abaixo:

1. Uma árvore saudável (valor 1) entra em chamas (valor 2) se houver ao menos uma árvore vizinha em chamas (valor 2).

2. Uma árvore em chamas (valor 2) se torna queimada (valor 3) após um ciclo de simulação e, a partir desse momento, não propagará mais o fogo.

3. A propagação do fogo ocorre exclusivamente nas direções ortogonais (esquerda, direita, acima e abaixo). A simulação deve considerar dois estados de propagação do fogo:

   (a) Sem influência do vento: o fogo se propaga linearmente nas quatro direções ortogonais.

   (b) Com influência do vento: o fogo se propaga apenas em direções ortogonais específicas, que devem ser configuradas no arquivo *config.h*. O vento pode ser configurado para direcionar o fogo para uma ou mais direções (esquerda, direita, acima ou abaixo).

4. O processo de propagação do incêndio continua até que não haja mais árvores em chamas (valor 2) ou que o número máximo de interações, $K$, seja alcançado.

A movimentação do animal na floresta segue as regras abaixo:

1. O animal, quando localizado em uma posição segura (valor 1 ou 0), deve procurar uma nova posição segura nas direções ortogonais. As posições seguras são classificadas da seguinte forma:

   (a) Melhor opção: posição com valor 4 (presença de água)

   (b) Opções intermediárias: posição com valor 0 (área vazia) e posição com valor 1 (árvore saudável).

   (c) Pior opção: posição com valor 3 (árvore queimada).

2. A posição do animal deve ser representada por uma variável externa à matriz, permitindo o seu acompanhamento ao longo da simulação.

3. Caso o animal se encontre em uma posição segura (valor 0), ele poderá permanecer nesta posição por até 3 interações, uma vez que tal posição não será afetada pela propagação do incêndio.

4. Quando o animal alcançar uma posição contendo água, essa célula deverá ser atualizada para o estado seguro (valor 0). Além disso, todas as posições ortogonalmente adjacentes deverão ser convertidas para o valor 1, simulando a dispersão da umidade e sua influência no controle da propagação do incêndio.

## 2 Requisitos do Trabalho

Esta seção apresenta as diretrizes essenciais para o desenvolvimento da solução, garantindo a adoção de boas práticas de programação e organização do código, bem como, da documentação.

## 2.1 Modularização do Código

- A solução implementada deve ser adequadamente modularizada, utilizando múltiplas classes/arquivos para melhor organização do código. O código resultante deve ser inserido na pasta do projeto, utilizando para isso uma subpasta chamada de **src**. Já o Makefile deve ser colocado fora da pasta src, contudo, dentro da pasta do projeto. A configuração do Makefile já está pronta para trabalhar nesse padrão.
- As declarações das classes e funções devem ser feitas em arquivos `.h` ou `.hpp`, enquanto as implementações devem estar nos arquivos `.c` ou `.cpp`.
- O arquivo `main.c` ou `main.cpp` não deve conter nenhuma implementação direta, apenas chamadas de métodos para execução da lógica elaborada.
- Toda compilação para testes e validação deve seguir os passos: (a) `make clean`; (b) `make`; (c) `make run`. Pequenas alterações podem não ser adequadamente realizadas se a diretriz `make clean` não for executada antes da compilação.

## 2.2 Entrada e Saída de Dados

O exemplo a seguir, que ilustra a entrada e a saída, assume a condição em que o vento não exerce influência sobre a propagação do fogo. Nesse cenário, o fogo se propaga exclusivamente para os lados ortogonais à sua origem, ou seja, para as direções cima, baixo, esquerda e direita. Além disso, indiferente a essa condição deve-se levar em conta:

- O programa deve ler a configuração inicial da simulação de um arquivo chamado `input.dat`.
- Os resultados da simulação deve ser gravado em um arquivo `output.dat`. Essa gravação deve ocorrer a cada finalização de interação.

**Exemplo de entrada (`input.dat`):**

```
5 5 1 1
1 1 1 1 4
1 2 1 1 1
1 1 1 1 4
0 0 1 1 1
1 4 1 0 4
```

Onde:

- `5 5` representam as dimensões da matriz ($N = 5, M = 5$).
- `1 1` indica a posição inicial do incêndio.
- O animal deverá iniciar sua posição em qualquer célula segura da matriz, ou seja, em uma posição cujo valor seja 0.
- As linhas seguintes representam a matriz inicial da floresta.

Considerando a matriz exemplificada acima, de dimensão $5 \times 5$, com o incêndio iniciando na posição $(1, 1)$, tem-se: (a) Como as matrizes em C/C++ utilizam indexação baseada em zero, essa posição corresponde exclusivamente à linha 1 e coluna 1 da matriz. Dado esse cenário, na primeira iteração da simulação, o fogo deverá se propagar ortogonalmente (acima, abaixo, à esquerda e à direita). Como resultado dessa interação, a matriz será atualizada conforme a seguinte representação:

**Saída (`output.dat`) após a primeira iteração:**

- (0,1) vira 2 (acima)
- (2,1) vira 2 (abaixo)
- (1,0) vira 2 (esquerda)
- (1,2) vira 2 (direita)

```
Iteração 1:
1 2 1 1 4
2 2 2 1 1
1 2 1 1 4
0 0 1 1 1
1 4 1 0 4
```

**Saída (`output.dat`) após a segunda iteração:**

Ao término da primeira iteração, os novos focos de incêndio nas posições $(0, 1)$, $(2, 1)$, $(1, 0)$ e $(1, 2)$ propagarão o fogo para seus vizinhos ortogonais na segunda iteração. Como resultado, a matriz será atualizada conforme a seguinte representação:

- (0,0) vira 2 (esquerda)
- (0,2) vira 2 (direita)
- (1,1) vira 3 (abaixo)
- (2,0) vira 2 (abaixo)
- (1,3) vira 2 (direita)
- (2,3) vira 2 (abaixo)

```
Iteração 2:
2 2 2 1 4
2 3 2 2 1
2 2 2 1 4
0 0 1 1 1
1 4 1 0 4
```

Além dos dados que registram a propagação do incêndio, é fundamental exibir, ao final da simulação, o caminho percorrido pelo animal, o número total de passos realizados e sua condição final, ou seja, se ele conseguiu sobreviver até o término da simulação.
Cada iteração da simulação impacta diferentes posições da matriz, tornando essencial a implementação de mecanismos para armazenar temporariamente essas alterações. Isso garante que o cálculo e a execução das interações subsequentes ocorram de maneira precisa e ordenada. Vale destacar que a propagação do incêndio deve sempre ocorrer após a movimentação do animal. Caso a posição atual do animal seja comprometida (i.e., atingida pelo fogo), aplica-se um mecanismo de *segunda chance*, permitindo-lhe uma tentativa imediata de fuga. Entretanto, essa oportunidade não será concedida se o animal estiver completamente cercado por árvores recém-incendiadas (valor 2), uma vez que não haveria rotas viáveis para a movimentação.

## 3 Funcionalidades Mínimas

Para que o trabalho seja considerado completo, ele deve implementar, no mínimo, os seguintes conceitos e funcionalidades:

- Implementação da floresta como uma matriz dinâmica de dimensão $N \times M$.
- Leitura do mapa inicial a partir de um arquivo *input.dat*.
- Definição de um ponto inicial para o fogo e executar a propagação do incêndio conforme as regras previamente estabelecidas.
- A simulação deverá ser implementada de forma visual e iterativa, prosseguindo automaticamente até que o incêndio se extinga ou que o limite de $K$ interações seja atingido. Nesse contexto, a abordagem visual-iterativa refere-se à exibição contínua e sequencial da evolução da simulação diretamente na tela do computador, permitindo ao usuário acompanhar o processo passo a passo. No entanto, o usuário não deve ter qualquer possibilidade de interferência na execução, não sendo solicitado a fornecer dados ou realizar interações mecânicas, como cliques ou confirmações, durante a simulação.
- Exibição do estado da floresta a cada iteração da simulação, com gravação do estado no arquivo de saída *output.dat*.
- Inclusão, na documentação (README.md), de uma análise sobre padrões de propagação do fogo. Para isso, apresentar exemplos visuais que ilustrem esses padrões, utilizando imagens. Elabore como parte dessa discussão o comportamento de ambas as simulações, sem a consideração do vento e com sua influência.
- Discussão sobre a capacidade da solução de simulação em prever o tamanho de um incêndio, considerando que cada interação pode ser interpretada como uma unidade de tempo.

- Discussão sobre possíveis algoritmos emergentes que poderiam contribuir com esse cenário, apresentando suas principais características e motivos que o levam a uma possível melhoria de desempenho.

# 4 Entrega do Trabalho

- O trabalho deverá ser entregue, unica e exclusivamente, por meio de um repositório no GitHub.
- O link do repositório deverá ser enviado no Moodle, em chamada do Moodle disponibilizada pelo professor.
- A data limite para envio é 20 de abril de 2025 até as 23:59h.

## 4.1 Estrutura do Repositório

O repositório deve conter:

1. **Código-fonte**: Arquivos *.c* ou *.cpp* organizados em módulos.
2. **Headers**: Arquivos *.h* ou *.hpp* para declarações de classes e funções.
3. **README.md**: Documentação explicativa do projeto.
4. **Arquivos de entrada e saída**: *input.dat* e *output.dat*.

# 5 Considerações Finais

Este trabalho tem como objetivo proporcionar uma experiência prática na implementação de algoritmos utilizando vetores e matrizes, estimulando a reflexão sobre a eficiência computacional. A proposta busca oferecer ao aluno a oportunidade de revisar e consolidar os conhecimentos adquiridos até o momento, reforçando os conceitos já abordados. Além disso, o trabalho visa aprimorar habilidades de pensamento crítico, desenvolvimento de soluções, aplicação de padrões de projeto e documentação adequada das ações envolvidas na resolução de problemas computacionais emergentes.

**Valor**: 15 pontos.