MAC0115 - Introdução à Computação para Ciências Exatas e Tecnologia

Alfredo Goldman e Beatriz Viana

Novembro de 2024

EP 3

Jogo da Vida

O Jogo da Vida é um autômato celular desenvolvido pelo matemático britânico John Horton Conway em 1970. Este jogo foi concebido com o objetivo de simular, de maneira simples, o comportamento de grupos de seres vivos e suas transformações ao longo do tempo. Suas regras geram padrões inesperados e, muitas vezes, belos, que variam desde padrões fixos até movimentos caóticos.

O Jogo da Vida tem várias aplicações, especialmente nas áreas de biologia, computação e teoria dos sistemas complexos, sendo frequentemente utilizado para estudar o comportamento emergente de sistemas dinâmicos.

Regras

O Jogo da Vida ocorre em um tabuleiro bidimensional infinito, onde cada célula pode estar em um dos dois estados: viva ou morta. As células interagem com suas oito vizinhas, que são as células adjacentes, tanto vertical e horizontalmente quanto nas diagonais. O estado do tabuleiro evolui de geração para geração de acordo com as seguintes regras:

1. Toda célula morta com exatamente três vizinhos vivos torna-se viva (nascimento).
2. Toda célula viva com menos de dois vizinhos vivos morre por isolamento.
3. Toda célula viva com mais de três vizinhos vivos morre por superpopulação.
4. Toda célula viva com dois ou três vizinhos vivos permanece viva.
5. As regras são aplicadas simultaneamente a todas as células do tabuleiro para determinar o estado da próxima geração.

Funções Prontas

Para facilitar o desenvolvimento do seu código, o código-fonte já disponibiliza algumas funções prontas:

- `gera_entrada_aleatoria(linhas::Int, colunas::Int)::MatrixInt`: Gera um estado inicial aleatório para o tabuleiro, com células vivas e mortas distribuídas de maneira aleatória com base no valor da variável `probabilidade`.

- `printa_matriz(matriz::MatrixInt)::Nothing`: Exibe o estado atual do tabuleiro na tela.

- `contabiliza_celulas(matriz::MatrixInt)::Int`: Conta o número de células vivas restantes no tabuleiro após as iterações.

Funções a Serem Desenvolvidas

O código também vem com algumas funções que precisam ser implementadas por você. As funções que devem ser desenvolvidas são:

- `atualiza_matriz(matriz::MatrixInt, debug::Bool = false)::MatrixInt`: Esta função deve tomar o estado atual do tabuleiro e atualizá-lo de acordo com as regras do Jogo da Vida.

- `main()::Nothing`: Esta função é responsável por ler a dimensão da matriz (linhas e colunas), a quantidade de iterações desejadas e verificar se o modo de debug deve estar ativado ou não. Caso o parâmetro `debug` seja igual a 0, apenas a primeira e a última iteração serão impressas. Se `debug` for igual a 1, todas as iterações serão exibidas.

  Além disso, a função `main()` também é responsável por chamar as demais funções do programa, gerando a matriz inicial e percorrendo todas as iterações pedidas.

  Caso todas as células morram antes de alcançar o número máximo de iterações, isso deve ser impresso na tela e o fluxo de atualizações deve ser interrompido.

  Essa função foi disponibilizada de forma parcial, apenas com a leitura dos dados e a inicialização da matriz, você deve implementar o restante de suas responsabilidades.

Exemplo de uso

Segue caso de uso do programa:

```
Digite o tamanho da matriz (linhas e colunas): 10 10
Digite o número de iterações: 5
(0) Sem debug (1) Com debug: 1

Estado inicial:
0 0 1 0 0 1 0 0 1 0
0 1 1 0 1 1 0 1 1 0
0 1 0 0 1 0 1 1 1 0
0 1 1 1 0 0 0 0 0 0
1 1 0 0 1 0 1 0 0 1
1 0 0 0 1 0 1 0 0 1
0 0 1 1 1 0 0 1 0 0
0 1 1 1 1 0 1 1 0 1
1 1 0 0 1 1 0 0 1 0
```

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

Matriz na 2º iteração:
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

Todas as células morreram na iteração 3.
Estado final:
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Entrega

Você deve usar o código-fonte fornecido como base para implementar as funções solicitadas. Após implementar sua solução, envie o arquivo com a extensão `.jl` (ou `.ipynb` se desenvolveu no Google Colaboratory) contendo o código completo. Certifique-se de que o código esteja bem estruturado, com boa indentação e comentários explicativos.

Além disso, ao final do código, inclua uma discussão sobre os valores de probabilidade para os quais há uma maior chance de a matriz estar com células vivas após 20 iterações. Realize simulações para uma matriz de tamanho 15 por 15 e verifique como diferentes valores de probabilidade influenciam o número de células vivas ao final da execução.

Referências

Aqui estão algumas referências que podem ser úteis para a compreensão do Jogo da Vida:

1. Mais informações sobre o Jogo da Vida.
2. Simulação do Jogo da Vida online.