

Programando em Java

(Matrizes)

Mestrado Integrado em Engenharia Informática FCT UNL

<http://ctp.di.fct.unl.pt/miei/ip/>

Corpo Docente 2020/2021

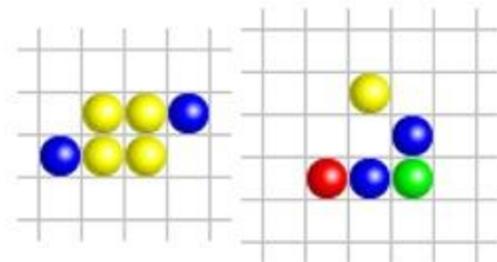
António Ravara, Artur Miguel Dias, Bernardo Toninho,
Ema Vieira, Inês Fernandes, Margarida Mamede,
Miguel Monteiro, Rui Nóbrega

Jogo da Vida (adaptado do ToPAS)

O Jogo da Vida não é um jogo tradicional. Não tem jogadores, nem se pode ganhar ou perder. Uma vez colocadas as peças, regras aparentemente simples determinam tudo o que vai acontecer, com comportamentos muito complexos, cheios de surpresas!

Joga-se num tabuleiro rectangular quadriculado. Cada quadrícula (ou célula) pode estar **morta** ou **viva**. Assinalamos uma célula viva com o carácter '#' e uma célula morta com o carácter '.'. Por exemplo, o tabuleiro inicial poderia ser:

```
. . . . .
. . . . .
. . . # . . .
. . . ### . . .
. . . # . # . . .
. . . # . . .
. . . . .
. . . . .
```



Jogo da Vida

Cada célula tem 8 células vizinhas (excepto as células no limite do tabuleiro), que são as quadrículas adjacentes em todas as direcções (incluindo as diagonais). Dada uma configuração inicial, o tabuleiro vai evoluindo com a aplicação das seguintes regras, que dependem do número de células vizinhas vivas:

- Uma célula morta com exactamente 3 vizinhas vivas nasce, ficando viva no passo seguinte;
- Uma célula viva com 2 ou 3 vizinhas vivas sobrevive, ficando viva no passo seguinte;
- Em todos os outros casos, a célula morre (se estava viva) ou mantém-se morta.

Estas regras são aplicadas ao mesmo tempo a todas as células, isto é, para determinarmos o que acontece num determinado passo temos de verificar o que acontece a cada célula antes de realmente fazermos as mudanças que as regras determinam.

Evolução do Tabuleiro

Tabuleiro inicial

Após 1 passo

.....
.....
... .# ##
... ## # . #
... # . # # . #
... # #
.....
.....

Como a primeira posição dos vectores é a 0, quando podemos escolher, é frequente considerar que:

- a primeira linha é a linha 0;
- a primeira coluna é a coluna 0.

Também é frequente numerar:

- as linhas de cima para baixo;
- as colunas da esquerda para a direita.

Célula (2,3) - está **morta**

Vizinhas vivas: **3**

em (2,4), (3,3) e (3,4)

Logo, nasce (estará **viva**)

Célula (3,3) - está **viva**

Vizinhas vivas: **3**

em (2,4), (3,4) e (4,3)

Logo, sobrevive (estará **viva**)

Célula (3,4) - está **viva**

Vizinhas vivas: **5**

em (2,4), (3,3), (3,5), (4,3) e (4,5)

Logo, morre (estará **morta**)

Evolução do Tabuleiro

Tabuleiro inicial

```

. . . . .
. . . . .
. . . # . . .
. . . ### . . .
. . . # # . . .
. . . # . . .
. . . # . . .
. . . . .
. . . . .
    
```

Após 1 passo

```

. . . . .
. . . . .
. . . ### . . .
. . . # # . . .
. . . # # . . .
. . . # # . . .
. . . # . . .
. . . # . . .
. . . . .
. . . . .
    
```

Após 2 passos

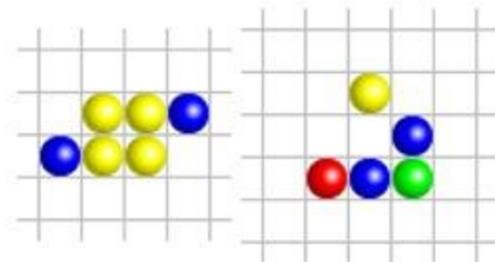
```

. . . . .
. . . # . . .
. . . # # . . .
. . # # . # # . .
. . # # . # # . .
. . # # . # # . .
. . # # . # # . .
. . # # . # # . .
. . # . . . .
. . . . .
. . . . .
    
```

Após 3 passos

```

. . . . .
. . . # . . .
. . # # . # # . .
. . # # . # # . .
. . # # . # # . .
. . # # . # # . .
. . # # . # # . .
. . # . . . .
. . . . .
. . . . .
    
```



Tarefa, Input e Output

Tarefa: Dada a configuração inicial de um tabuleiro, calcular o estado do tabuleiro após um determinado número de passos.

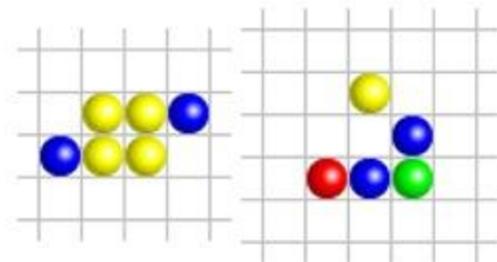
Forma do Input

L C P *L* (nº de linhas) e *C* (nº de colunas) são inteiros em [1, 100]
linha_1 *P* (nº de passos) é um inteiro em [1, 250]
linha_2
.....
linha_L

Cada linha é uma sequência com *C* caracteres;
cada carácter é '#' ou '.'

Forma do Output

linha_1
linha_2
.....
linha_L



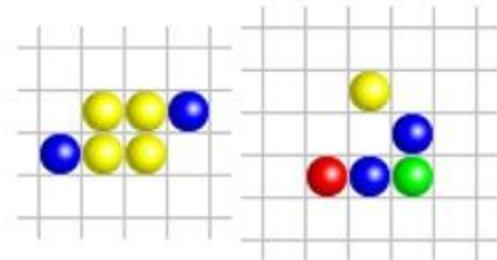
Exemplo

Input

```
8 9 3
. . . . .
. . . . .
. . . . .
. . . # . . .
. . . ### . . .
. . . # # . . .
. . . # . . .
. . . # . . .
. . . . .
. . . . .
```

Output

```
. . . . .
. . . # . . .
. . ## . ## . .
. . ## . ## . .
. . ## . ## . .
. . . # . . .
. . . . .
. . . . .
```



Classes e Questões de Implementação

Classes

- **GameOfLife**: responsável pelo estado do tabuleiro
- **Main**: responsável pela leitura dos dados e pela escrita dos resultados

Questões

1. Como guardar o estado do tabuleiro, sabendo que cada célula está viva ou morta e, em cada passo, é necessário contar o número de vizinhas vivas de cada célula?
2. Como é que a classe Main “passa” à classe GameOfLife o tabuleiro inicial?
3. Como é que a classe Main “recebe” da classe GameOfLife o tabuleiro final?

Como Guardar o Tabuleiro?

1. Cada célula está viva ou morta \longrightarrow `boolean[][]`
2. É necessário contar o número de vizinhas vivas de cada célula.

Quantas células vizinhas tem uma célula? Será possível uniformizar?

3				
				5
	8			

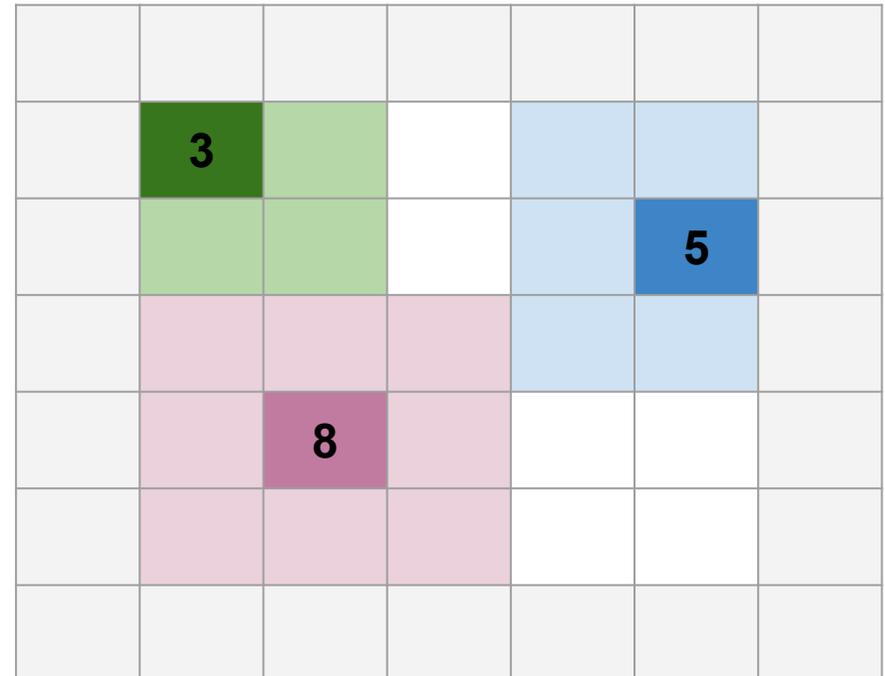
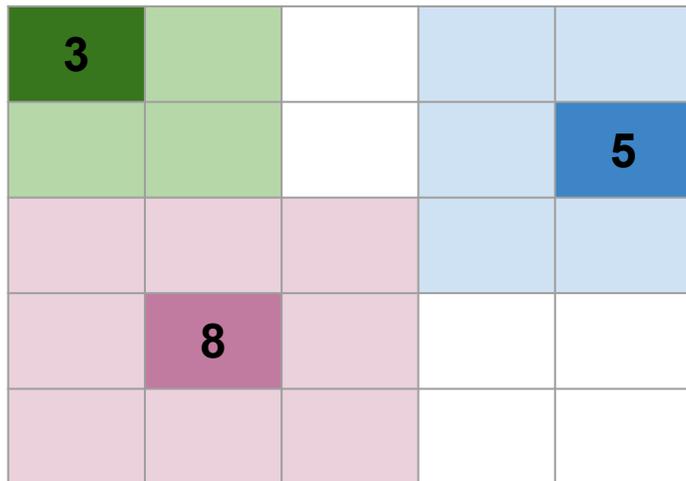
Como Guardar o Tabuleiro?

1. Cada célula está viva ou morta \longrightarrow `boolean[][]`
2. É necessário contar o número de vizinhas vivas de cada célula.

Quantas células vizinhas tem uma célula? Será possível uniformizar?

As células cinzentas estão mortas e não mudam de estado.

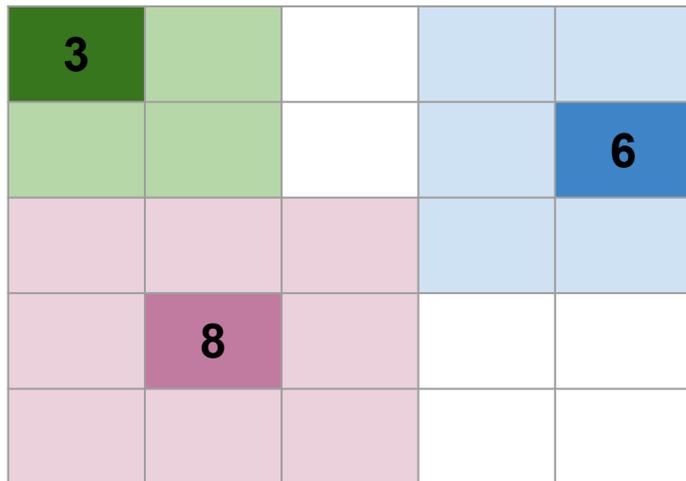
célula (i, j) \longrightarrow célula $(i+1, j+1)$



Main e GameOfLife: visões \neq do tabuleiro

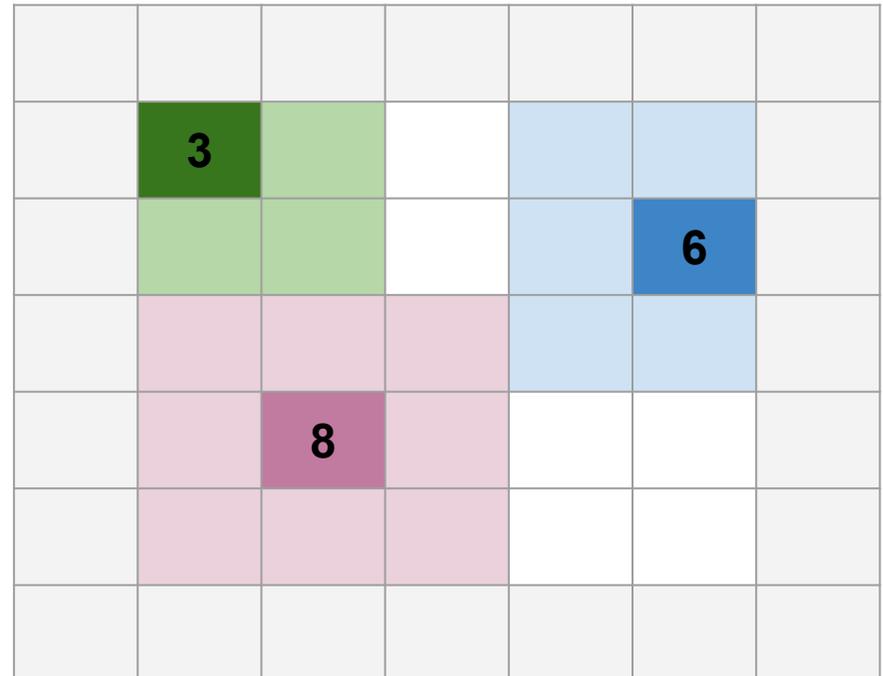
Para a Main:

- o tabuleiro tem L linhas e C colunas
- '#' e '.' são os códigos para célula viva e célula morta



Para a GameOfLife:

- o tabuleiro tem $L+2$ linhas e $C+2$ colunas
- `true` e `false` são os códigos para célula viva e célula morta



Tabuleiro Inicial: Main → GameOfLife

Como é que a classe Main “passa” à classe GameOfLife o tabuleiro inicial?

- Depois de ler `L` e `C`, a Main pode criar o jogo: `GameOfLife(L, C)`

```
// Pre: rows > 0 && cols > 0
public GameOfLife( int rows, int cols ) {
    // A matriz de booleanos é criada
    // com rows+2 linhas e cols+2 colunas.
}
```

- O tabuleiro inicial pode ser comunicado célula a célula:

```
// Pre: row >= 0 && row < rows && col >= 0 && col < cols
public void addCell( int row, int col, boolean isAlive ) {
    // Atribui-se o valor de isAlive à célula da matriz
    // de booleanos na linha row+1 e na coluna col+1.
}
```

Tabuleiro Final: GameOfLife → Main

Como é que a classe Main “recebe” da classe GameOfLife o tabuleiro final?

- O tabuleiro final pode ser construído com L linhas e C colunas, para satisfazer o pedido, e devolvido:

```
public boolean[][] getState( ) {  
    // Cria-se uma nova matriz de booleanos,  
    // com L linhas e C colunas,  
    // copiam-se as células do “tabuleiro real” e  
    // devolve-se a nova matriz.  
}
```

Notas:

- Um tabuleiro não é uma sequência de células. Não faria sentido devolver um iterador das células presentes no tabuleiro.
- Devolve-se uma cópia (de uma parte) do tabuleiro guardado. Portanto, só a classe GameOfLife pode alterar os seus tabuleiros.

GameOfLife (sem os métodos privados)

```
public class GameOfLife {  
  
    private boolean[][] state;        // L+2 linhas, C+2 colunas  
    private int rows;                 // rows guarda L  
    private int cols;                 // cols guarda C  
  
    // Pre: rows > 0 && cols > 0  
    public GameOfLife( int rows, int cols ) {...}  
  
    // Pre: row >= 0 && row < rows && col >= 0 && col < cols  
    public void addCell( int row, int col, boolean isAlive )  
    {...}  
  
    public boolean[][] getState( ) {...}  
  
    // Pre: ticks >= 0  
    public void evolve( int ticks ) {...}  
}
```