Universidade do Porto
Faculdade de Engenharia
FEUP

Relatório Intercalar

Lear

Faculdade de Engenharia da Universidade do Porto

Programação em Lógica
Professor Rui Camacho

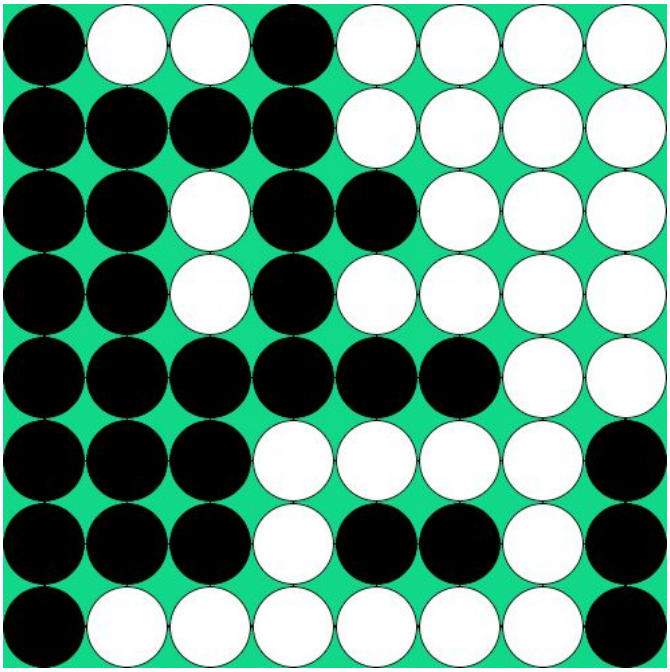
2017/2018

Trabalho Realizado por:

Alexandre José da Silva Carvalho- up201506688
Vitor Emanuel Fernandes Magalhães te- up201503447

Descrição do Jogo

Enfastiado pela obrigação de capturar peças no jogo Othello e inspirado neste último, Luís Bolaños Mures decidiu criar o Lear.



Regras

O jogo é jogado por duas equipas, a equipa Preta e a equipa Branca.
A equipa preta joga primeiro.
Em cada turno, o jogador tem de colocar uma peça da sua cor num espaço vazio.
Se, ao colocar a peça, constituir uma linha não interrompida de peças que contém:

a) Duas peças da sua cor e
b) Uma linha ininterrupta de peças inimigas.

As peças inimigas viram, ou seja, são substituídas por peças da cor do jogador a jogar.
As duas peças da mesma cor mencionadas poderão estar no início da linha ou no fim da linha, tanto juntas como uma em cada lado.
O jogo termina quando o tabuleiro estiver cheio. O vencedor é o jogador com mais pontos.
Os pontos são calculados pelo número de peças mais o valor do komi, usado quando apropriado.

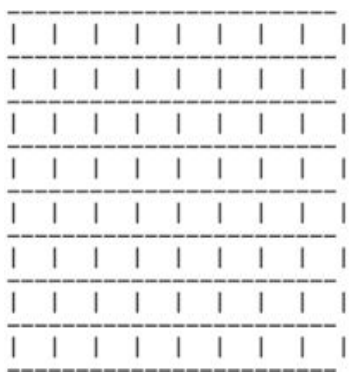
O komi é um valor adicionado ao número de pontos do jogador que não fez a última jogada. Este valor depende do tamanho do tabuleiro. Se o tabuleiro é ímpar, o komi será par e adicionado à equipa Branca. Caso contrário, será ímpar e adicionado à equipa Preta.
O valor do komi é indicado pelo primeiro jogador. O segundo jogador escolhe a sua cor.

Referências:
https://boardgamegeek.com/thread/1633900/new-game-lear
https://boardgamegeek.com/boardgame/209777/lear

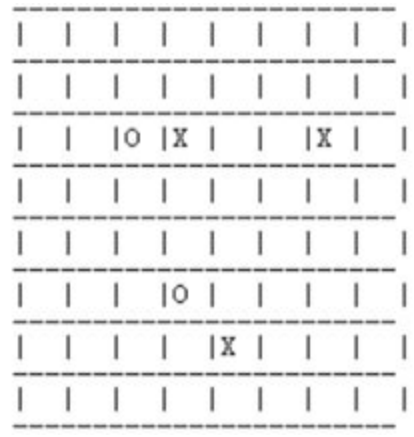
Representação do Estado de Jogo

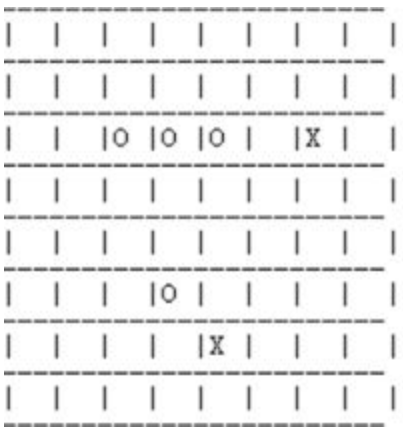
O estado de jogo é representado por uma lista de listas. As peças Brancas são representadas por um 'O' e as peças Pretas por um X.

Exemplo de Tabuleiro Inicial:



Exemplo de Tabuleiro Intermédio antes e depois da Equipa Branca jogar:





Como jogou ao lado do X e formou uma linha com duas peças brancas e uma peça preta, a peça preta tornou-se branca.

Exemplo de Tabuleiro Final:

```
-------------------------
|O |O |O |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
|X |X |X |X |X |X |X |X |
-------------------------
```

Neste exemplo, as peças pretas claramente ganharam.

Visualização do Tabuleiro

Para criar um tabuleiro, é usado o predicado initialBoard, como demonstrado abaixo:

```
initialBoard([
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
	[emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell]]).
```

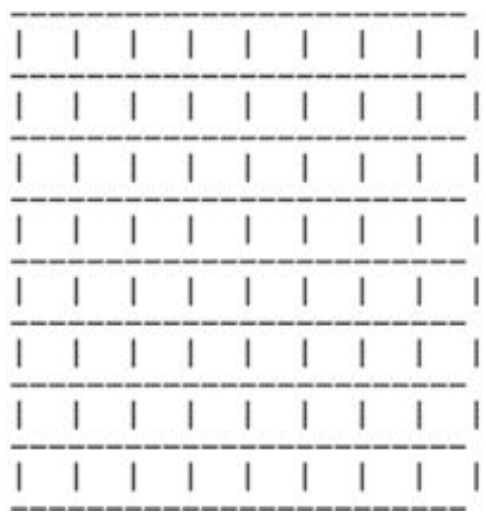
O valor emptyCell representa uma célula vazia.

Para imprimir o tabuleiro no ecrã, são utilizados os seguintes predicados:

```
printBoard([]).
printBoard([Head|Tail]) :-
	write('|'),
	printRow(Head),
	printRowSeparator,
	printBoard(Tail).

printRow([]) :- nl.
printRow([emptyCell|Tail]) :-
	write('  '),
	write('|'),
	printRow(Tail).
```

```
printRow([Head|Tail]) :-
        Head \= emptyCell,
        write(Head),
        write('|'),
        printRow(Tail).

printLineSeparator(1):- write('|'), nl.
printLineSeparator(NLines):-
        write('|'),
        nl,
        Next is NLines-1,
        printLineSeparator(Next).

printRowSeparator:-write('------------------------'), nl.
```

O output destes dois predicados será o representado abaixo:



Movimentos

Cada jogador poderá colocar uma peça numa célula vazia do tabuleiro.
Os predicados que serão utilizados são:
-> verifyMovement;
-> getPiece;
-> getElemPos;
-> setPiece;
-> setNLine;
-> setNColumn;