Universidade Federal Fluminense

Instituto de Ciência e Tecnologia

Departamento de Computação

Trabalho sobre o jogo Reversi (primeira parte)

O jogo Reversi (também conhecido como Othello) é um jogo de tabuleiro para dois jogadores. O tabuleiro utilizado é de tamanho 8x8 e em cada casa são colocadas “pedras” pretas ou brancas (cada pedra tem um lado preto e o outro lado branco). Inicialmente, o tabuleiro é montado como na Figura 1 abaixo, no que é chamado de Posição Inicial.

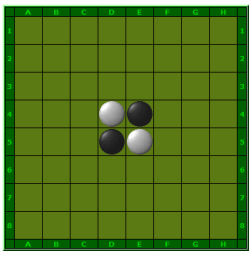


Figura 1: posição inicial do tabuleiro de Reversi

O primeiro jogador a fazer sua jogada é o jogador de pretas. A jogada consiste em colocar uma peça (da cor do jogador) em alguma casa vazia do tabuleiro, de maneira a fazer um “sanduíche” com pelo menos uma peça do adversário. Esse “sanduíche” consiste em uma ou mais peças do adversário, em linha reta (vertical, horizontal ou diagonal), que estejam entre uma pedra já colocada anteriormente e a pedra que acabou de ser colocada. A Figura 2 mostra uma posição do jogo e indica com X vermelho onde o jogador de pretas pode colocar uma pedra para fazer um ou mais sanduíches.

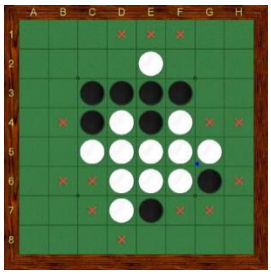
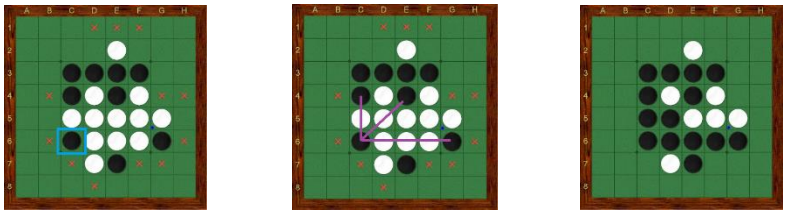


Figura 2: posição intermediária do jogo (o jogador de pretas pode colocar uma pedra em qualquer casa marcada com X vermelho).

Todas as peças que estejam nesses sanduíches são “viradas”, ou seja, passam a pertencer momentaneamente ao adversário, como nas Figuras 3. Todas as pedras do adversário que puderem ser viradas, por estarem em um sanduíche, devem ser viradas imediatamente após a colocação da pedra. Não é possível escolher quais

pedras serão viradas nem deixar pedras sem serem viradas em quaisquer direções possíveis. Uma pedra não pode ser colocada em uma posição onde não vire nenhuma pedra do adversário.



A **B** **C**

Figuras 3: posições intermediárias do jogo. (A) pedra colocada pelo jogador de pretas na casa marcada pelo retângulo azul. (B) todos os sanduíches possíveis com a colocação da pedra são indicados pelos segmentos de reta roxos. (C) a posição intermediária após as pedras brancas (dos sanduíches) serem viradas.

Após colocar uma peça e "virar" as peças do adversário. O jogador passa a vez para o adversário. Se, a qualquer momento, o jogador da vez não puder fazer uma jogada, ele passa a vez para o adversário sem colocar nenhuma pedra. Se os dois jogadores não puderam mais colocar pedras, o jogo termina. O joga também termina quando todas as casas do tabuleiro estiverem com alguma pedra. Ao final, vence o jogador que terminar com mais pedras. Se existirem casas vazias ao final da partida, elas são somadas à pontuação do jogador vencedor.

O que deve ser implementado?

1) **O tabuleiro deve ser representado por uma matriz 8x8 de números inteiros. As casas vazias possuem o valor 0. As pedras brancas são representadas pelo valor 1 e as pedras pretas pelo valor -1.**
2) **Cada jogada é representada pelo struct jogada (indicado abaixo).**
3) **Faça uma função IniciaTabuleiro(), que receba uma matriz como parâmetro e retorne o tabuleiro na posição inicial.**
4) **O programa deve ter uma função DesenhaTabuleiro(), que receba o tabuleiro (matriz) e desenhe na tela as pedras já colocadas no tabuleiro da melhor forma possível (parecendo uma matriz na tela). As peças pretas ("black") devem ser mostradas por uma letra B maiúscula. As peças brancas ("white") devem ser mostradas por uma letra W maiúscula. A função também deve indicar na tela a numeração das linhas e colunas.**
5) **Faça uma função EscolheJogada(), na qual o jogador da vez indique uma linha e uma coluna onde ele deseja colocar sua pedra. A jogada (struct jogada) deve ser retornada pela função.**
6) **Faça uma função ExecutaJogada() que recebe o tabuleiro (matriz), o jogador da vez e a jogada (struct jogada) retornada pela função EscolheJogada(). Se a jogada for válida, a função deve modificar o tabuleiro de acordo com as regras e retornar 1; caso contrário, deve manter o tabuleiro como está e retornar 0.**
7) **Se a função ExecutaJogada() retornar 0, o programa deve indicar que a jogada é inválida e solicitar uma nova jogada. Isso deve acontecer até que o jogador da vez informe uma jogada válida. Considere, nessa primeira versão do trabalho, que sempre existirá pelo menos uma jogada valida.**
8) **Se a função ExecutaJogada() retornar 1, o programa deve mudar o jogador da vez.**

9) **Quando nenhuma casa estiver vazia, o jogo deve informar quem foi o jogador vencedor ou se houve empate.**

O que não é necessário implementar (ainda)?

1) **A situação onde não há jogadas válidas disponíveis para o jogador da vez.**

```
struct jogada{

int linha,coluna;

}
```