RELATÓRIO

Este relatório aborda os **Bots** e as funções importantes **“valida”** e **“jogada”** do **Projeto Prático de Laboratório de Algoritmia**, o jogo **“Reversi”**.

Este relatório é composto por duas partes, a parte 1 onde iremos abordar as funções “valida” e “jogada” e a parte 2 onde iremos abordar os Bots.

Parte 1 - valida e jogada:

Nesta parte do relatório iremos abordar o funcionamento de duas das funções mais importantes deste projeto, a “valida” e a “jogada”.

valida:

O objetivo desta função é validar uma jogada dando return a um int (0 ou 1) dependendo se a jogada é inválida ou válida respetivamente.

Esta função recebe **4 argumentos**, sendo eles o **estado** atual, as **coordenadas da jogada (x,y)** e o **player** que está atualmente a jogar (‘X’ ou ‘O’);

A função é repartida em **8 funções auxiliares**, uma para cada direção, para **cima, baixo, esquerda, direita e as 4 diagonais**, estas funções auxiliares também dão return a um int (0 ou 1) que significa que a jogada é inválida ou válida;

Para uma jogada ser valida numa direção, tem que existir pelo menos uma peça inimiga nessa direção e pelo menos uma peça igual à que queremos jogar logo a seguir as peças inimigas, para assim as peças do adversário serem “encurraladas”, logo existe no máximo 6 peças do adversário que conseguimos encurralar numa direção.

Cada uma dessas funções auxiliares começa na “casa” onde vai ser jogada a peça (em caso de ser uma jogada valida), e vai verificar se a peça na “casa” imediatamente a seguir (na respetiva direção) é o inverso da que queremos jogar .

Em caso de não ser, a função da return 0, em caso de ser, começa um ciclo em que se a peça na “casa” imediatamente a seguir for igual ao inverso da que queremos, avançamos para a próxima “casa” (na respetiva direção).

Este ciclo só para quando a peça na "casa” a ser avaliada é diferente do inverso da queremos jogar ou já foram avaliadas 6 “casas” , no fim do ciclo avaliamos a peça na “casa” em que o ciclo acabou e verificamos se é igual à peça que queremos jogar , se for quer dizer que podemos “encurralar” peças inimigas por isso é um jogada valida , caso contrario da return 0.

Fazendo este procedimento para as 8 direções a função valida vai dar return 1, em caso de qualquer uma das funções auxiliares dar return 1.

jogada:

O objetivo desta função é jogar nas coordenadas recebidas e mudar as peças que devem ser mudadas (as que ficaram encurraladas) dando return a um Estado.

Esta função tem um funcionamento parecido ao da valida, tem 8 direções para qual ela vai mudar as peças, cada direção só vai ser chamada se a valida auxiliar que corresponde a essa direção der return a 1;

Assim cada direção da jogada vai mudar a peça que está naquela posição para a mesma peça de quem jogou, depois de mudar vai incrementando ou decrementando os índices de acordo com a direção que foi atribuída, repetindo o método apenas parando quando encontrar uma peça igual á que foi jogada.

Parte 2 - BOTS:

Nesta parte iremos abordar a diferença entre os níveis dos bots, a função usada e o seu funcionamento e os argumentos recebidos por esta.

Diferença entre os níveis:

Os nossos bots foram desenvolvidos em 3 níveis diferentes **fácil**, **médio** e **difícil**, correspondendo aos números de 1-3 respetivamente;

O **bot fácil** apenas joga na primeira jogada válida, percorrendo assim todo o tabuleiro por linhas da esquerda para a direita, até encontrar uma jogada válida, jogando nessa posição;

Para os **bots médio e difíci**l usamos o método **MiniMax.**

-> Usando o algoritmo **Alpha–beta pruning** para diminuir o número de nodos avaliados pelo **MiniMax**;

-> Assim demos um **valor a cada posição do tabuleiro de acordo com a vantagem estratégica** que cada posição oferece, variando estes valores de [-99,99];

-> Geramos assim uma **árvore de tabuleiros válidos** de acordo com as jogadas válidas para o bot e calculamos quanto valia cada tabuleiro gerado a partir de cada jogada válida;

Para o **bot médio** geramos uma árvore de **altura 1**, ou seja, apenas os tabuleiros gerados pelas jogadas válidas do bot, assim o bot vai jogar na jogada correspondente ao tabuleiro mais vantajoso.

Para o **bot difícil** geramos uma árvore de **altura 3**, ou seja, geramos primeiro os tabuleiros para as jogadas válidas do bot, depois para as do jogador e de seguida para o bot jogando na posição mais favorável.

Função usada:

Criamos uma **função** usada para os bots médio e difícil é chamada **minimax.**

Esta função recebe **8 argumentos** e funciona **recursivamente** sendo chamada tantas vezes como a altura dá arvore escolhida.

A função tem como **return** um **int** que é os pontos da **melhor jogada atualmente encontrada.**

-> Os argumentos recebidos pela função são:

i. O estado atual do jogo

ii. A altura até qual vai pesquisar as jogadas

iii. **2 ints "alpha" e "beta"** que servem para **controlo do pruning** evitando pesquisas desnecessárias **ALPHA** iniciado em **(-infinito)** que é a **pior jogada possível para o bot** e o **BETA** iniciado em **(+infinito)** que é a **melhor jogada possível para o bot** e são alteradas, o ALPHA pelo **máximo** entre o **tabuleiro atual** e o **ALPHA guardado** e o **BETA** no mesmo sentido pelo **mínimo entre o tabuleiro atual** e o **mínimo guardado em beta** se a condição (BETA <= ALPHA) é satisfeita a pesquisa de tabuleiros mais profundos é parada nesse "ramo" ou seja ocorre o pruning.

iv. Um char que chamamos de maximizante que se for 'T' vamos contar a pontuação de forma positiva e se for 'F' de forma negativa

v. 2 Apontadores xx e yy que será a melhor jogada onde o bot irá jogar

vi. E a peça com que o BOT está a jogar