UNIVERSIDADE DE ÉVORA

Trabalho Final IA

OURI

Trabalho realizado por:

Marcelo Feliz nº 38073

Ruben Bravo nº 37548

1.Introdução

Neste trabalho o objetivo é criar duas IA's que consigam jogar Ouri, tanto contra 1 jogador como entre as duas.

O Ouri é um jogo de tabuleiro para duas pessoas, e cada jogador tem 6 casas ou buracos e 1 depósito. No inicio do jogo cada buraco tem 4 sementes, ou seja, existem 48 sementes no total. O objectivo do jogo é recolher mais sementes que o adversário e vence o jogador que obtiver 25 ou mais sementes.

Regras:

O tabuleiro é composto por duas filas de seis buracos aos quais se chamam casas e dois buracos maiores nas extremidades designados por depósitos. Estes depósitos servem para colocar as sementes capturadas ao adversário ao longo do jogo. Cada jogador escolhe o seu lado do tabuleiro. Para decidir quem começa, um dos jogadores esconde uma semente numa das mãos, se o outro adivinhar correctamente em que mão está começa o jogo. Os jogadores sentam-se frente a frente e o depósito que lhe pertence é o que está à sua direita.

Movimentos:

No início do jogo são colocadas 4 sementes em cada uma das doze casas.

O jogador que abre o jogo colhe todas as sementes de um dos seus buracos e distribui uma a uma nos buracos seguintes, no sentido anti-horário (sentido contrário ao do ponteiro do relógio). Esta regra mantém-se para todas as jogadas.

Se a casa contiver mais do que 12 sementes, o jogador dá uma volta completa ao tabuleiro e saltando a casa donde partiu.

Capturas:

A captura é a última parte do movimento: Se ao depositarmos a última semente numa casa do adversário e esta contenha duas ou três sementes (contando com a semente que acabámos de depositar), podemos capturá-las. Isto é, retiramos todas as sementes e guardamo-las no nosso depósito. Sempre que as casas anteriores à última tiverem duas ou três sementes e pertençam ao adversário podemos e devemos capturá-las, até que encontremos uma casa que não cumpra alguma destas condições.

Se ao depositarmos a última semente numa casa do adversário e esta contenha quatro ou mais sementes (contando com a semente que acabámos de depositar), não podemos capturá-las. Passando o jogo para o adversário.

Regras suplementares:

As regras suplementares aplicam-se quando um dos jogadores fica sem sementes:

Se ao realizar um movimento o jogador fica sem sementes o adversário é obrigado a efectuar um movimento que introduza sementes no seu lado;

Se um jogador realizar uma captura e deixar o adversário sem sementes, este (jogador que efectuou a captura) vê-se obrigado (no caso de ter sementes que o permitam) a jogar de forma a introduzir sementes nas casas do adversário. Nota: Se o jogador não tiver sementes finaliza a partida**.**

Fim da Partida:

Quando um jogador capturar a maioria das sementes ,25 ou mais a partida finaliza e esse jogador ganha.

Quando um jogador fica sem sementes e o adversário não pode jogar de forma a introduzir sementes nas casas deste jogador, a partida termina e o adversário recolhe as sementes que estão nas suas casas para o seu depósito. Ganha quem tiver um maior número de sementes.

Quando a partida está a finalizar e ficam poucas sementes no tabuleiro criando uma situação que se repete ciclicamente, sem que os jogadores possam ou queiram evitá-lo, cada jogador recolhe as sementes que se encontram nas suas casas e colocam-nas nos respectivos depósitos. Ganha quem tiver mais sementes.

2. Ouri

Decidimos implementar o algoritmo minimax para a 1ª IA porque queríamos descobrir qual a "melhor" jogada para uma determinada situação do tabuleiro do jogo, sendo ela qual for. Ao escolhermos o minimax, como temos a informação perfeita sobre o estado do tabuleiro podemos descobrir qual a melhor jogada naquele momento. A linguagem de programação utilizada foi o python.

O algoritmo minimax é utilizado em teoria de decisão e teoria do jogo para encontrar a melhor jogada para um determinado jogador, assumindo que o oponente irá jogar de forma otimizada ou seja também jogando a melhor opção naquela altura. Este algoritmo é utilizado em jogos como o jogo do galo, xadrez, etc. Neste algoritmo os jogadores são chamados de maximizer e de minimizer, tendo como objectivo conseguir a máxima pontuação e a mínima respectivamente.

Pseudocódigo no qual a implementação foi baseada:

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

A 2ªa IA só tem 1 nível de profundidade e tenta sempre jogar a melhor jogada para cada altura, ou seja, aquela que lhe permite comer mais peças não pensando nas próximas jogadas, nem nas suas nem nas do adversário. Se não existir uma melhor jogada, escolhe entre todas as possíveis uma, aleatoriamente. Esta implementação foi feita por nós sem qualquer recurso além das regras do jogo, tendo também sido implementada em python. Em vez desta IA poderíamos ter implementado uma versão melhorada do minimax ou outro algoritmo mas não existia muita documentação sobre o assunto.

Em teoria o algoritmo da IA1 deveria ganhar sempre ou quase sempre contra a IA2, no entanto percebemos que ao usar só 1 tabuleiro auxiliar não é o suficiente para que o minimax teste todas as opções de jogadas possíveis. No entanto não conseguimos resolver este problema já que era necessário criar um número indefinido de tabuleiros auxiliares.

No algoritmo da 1ª IA, usando o minimax, para um tempo de resposta de 5 segundos o depth será 8, para um tempo de resposta de 15 segundos o depth será 9 e para um tempo de 30 segundos seria 10.

Para correr o programa na linha de comandos basta escrever python3 ia.py .

Através do seguinte menu escolhemos quem joga primeiro, se a IA ou o jogador.