UNIVERSIDADE DE ÉVORA

Engenharia Informática
Inteligência Artificial

Trabalho Final - Jogo Ouri

Professor: Paulo Quaresma

Sarah Simon Luz 38116
Ana Ferro 39872

24 Junho, Ano Letivo 2019/2020

Índice

1 Jogo Ouri

O jogo Ouri é um jogo de tabuleiro jogado por 2 jogadores. O tabuleiro é constituído por duas filas de 6 posições ("buracos"), perfazendo no total de 12 buracos, e dois outros buracos maiores posicionados em cada extremidade do tabuleiro ("depósitos") que servem para armazenar as peças capturadas.

O objetivo do jogo é recolher mais peças que o adversário, vence aquele que obtiver 25 ou mais peças.

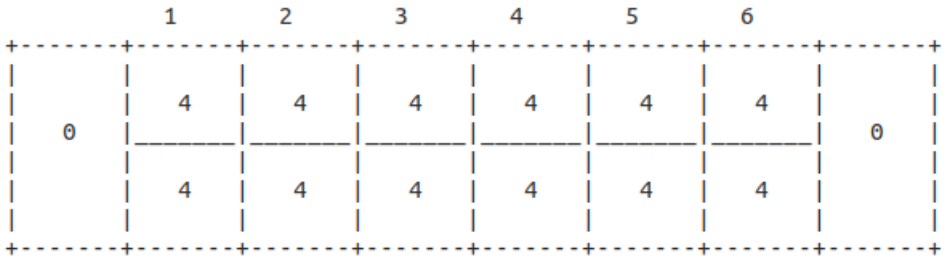

Figura 1: Tabuleiro inicial

1.1 Regras

1.1.1 Movimentos

O jogo começa com 4 peças em cada um dos 12 buracos, estando em jogo um toral de 48 peças.

O jogador colhe todas as peças do buraco que escolheu e vai distribuindo peça a peça nos buracos seguintes no sentido contrário ao relógio. Se o buraco que escolheu tiver mais que 12 sementes, dá uma volta completa ao tabuleiro e salta o buraco de onde partiu.

Só é permitido mexer em buracos com 1 peça se não houver outros com mais.

1.1.2 Capturas

Ao depositar a última peça num buraco do adversário com 2 ou 3 peças, incluindo a peça que depositámos, podemos captura-las e colocá-las no depósito correspondente. Este mecanismo de captura repete-se nos buracos anteriores até não se cumprir essas condições.

1.1.3 Regras suplementares

Quando um dos jogadores fica sem sementes, o adversário é obrigado a efetuar um movimento que meta peças no outro lado.

- Se isso não for possível, o jogo chega ao fim e o adversário recolhe as restantes peças para o seu depósito.

1.1.4 Fim do jogo

O jogo chega ao fim quando:
- Um dos jogadores capturar 25 ou mais peças.
- Chegamos a um empate, cada jogador capturou 24 peças.
- Um dos jogadores fica sem peças e o adversário não consegue introduzir peças do lado do jogador.
- Detetamos um ciclo, algo que os jogadores não conseguem evitar. Neste caso, cada jogador recolhe as suas peças.

2 Algoritmos

Por estarmos perante um jogo de 2 jogadores, com conhecimento do ambiente, nomeadamente qual o objetivo do jogo e das jogadas possíveis, optamos por escolher algoritmos que consigam pensar no desenvolvimento do jogo de acordo com a jogada que o adversário faz ou vai fazer.

Assim sendo, para este trabalho, implementamos o *Algoritmo Minimax* e o *Algoritmo Minimax com corte Alfa-Beta* em Python.

2.1 Minimax

O objetivo do *Minimax* é minimizar a perda máxima possível, ou, maximizar o ganho mínimo, visando conseguir chegar á melhor jogada.

1. Construímos a árvore até alcançar a profundidade definida ou chegar ao estado terminal.
2. Vai se subindo nos nós e dependendo de quem é a vez de jogar (jogador ou oponente), guardamos o resultado.

2.1 Caso for a vez do jogador, o algoritmo guarda o maior (*maximizer*) resultado das suas respetivas ramificações.

2.2 Caso for a vez do oponente, o algoritmo guarda o menor (*minimizer*) resultado das suas respetivas ramificações.

3. Este processo repete-se até chegar á raiz.

2.2 Minimax com corte Alfa-Beta

É uma variação do algoritmo Minimax.

Permite reduzir o número de nós que são avaliados ao eliminar uma ramificação da árvore de pesquisa de forma a que não seja examinada (pois é desnecessário) e com isso acelerar o processo de pesquisa.

Daí conseguirmos fazer uma pesquisa até uma profundidade maior para o mesmo nível de análise, comparado com o Minimax.

3 Níveis de Análise

Deparamo-nos com alguns problemas na implementação desta parte do trabalho. Primeiro tentamos com que o algoritmo parasse de pesquisar ao fim de x segundos, mas não conseguimos. A alternativa foi estudar o comportamento dos algoritmos mediante a profundidade e pré defini-las.

Reparamos que á medida que o jogo vai progredindo, o tempo para calcular a melhor jogada tem tendência para diminuir. Assim, a primeira jogada (partindo do estado inicial de um jogo normal) vai ser aquela que demorará mais tempo. Foi com base nessa primeira jogada que decidímos qual a profundidade usar.

Outra observação que fizemos foi que, os algoritmos têm uma *performance* melhor quando a profundidade é um número ímpar, mas por questões de cumprir os tempos estabelecidos tivemos que estabelecer algumas profundidades pares.

3.1 5 segundos

- Minimax
  - Profundidade: 7
- Alfa-Beta
  - Profundidade:11

3.2 15 segundos

- Minimax
  - Profundidade: 9
- Alfa-Beta
  - Profundidade:13

3.3 30 segundos

- Minimax
  - Profundidade: 10
- Alfa-Beta
  - Profundidade:15

Tal como foi referido na secção anterior, o *algoritmo Minimax com corte Alfa-Beta* consegue fazer uma pesquisa mais profunda para o mesmo nível de análise daí a diferença de profundidades.

4 Torneio

```
1º jogo

                Tempo: 5segundos
                1ºJogador: Alfa-Beta
                2ºJogador: Minimax
                Vencedor: Alfa-Beta (25-14)

2º jogo

                Tempo: 5segundos
                1ºJogador: Minimax
                2ºJogador: Alfa-Beta
                Vencedor: Minimax (27-21)

3º jogo

                Tempo: 15segundos
                1ºJogador: Alfa-Beta
                2ºJogador: Minimax
                Vencedor: Minimax (13-27)

4º jogo

                Tempo: 15segundos
                1ºJogador: Minimax
                2ºJogador: Alfa-Beta
                Vencedor: Alfa-Beta (13-27)

5º jogo

                Tempo: 30segundos
                1ºJogador: Alfa-Beta
                2ºJogador: Minimax
                Vencedor: Alfa-Beta (31-8)

6º jogo

                Tempo: 30segundos
                1ºJogador: Minimax
                2ºJogador: Alfa-Beta
                Vencedor: Alfa-Beta (6-25)
```

Os resultados são os esperados, o Alfa-Beta ganha a maioria dos jogos. Isto porque, para o mesmo nível de análise, este alcança uma profundidade maior, o que se torna uma vantagem sobre o Minimax. O 3ºjogo e o 4ºjogo levantam a questão de se a ordem pela qual se joga influencia o desfecho do jogo, pois em ambos os jogos cada algoritmo ganhou com o mesmo resultado, no entanto, não chegámos a nenhuma conclusão quanto a isso.

5 Como correr o programa?

Para correr:

python3 Ouri.py -p ou ***python3 Ouri.py -s***

Os parâmetros *-p* e *-s* indicam se o programa joga em primeiro ou segundo lugar. No caso do torneio, indica se o *Minimax* joga primeiro ou não.

De seguida deparamo-nos com o menu:

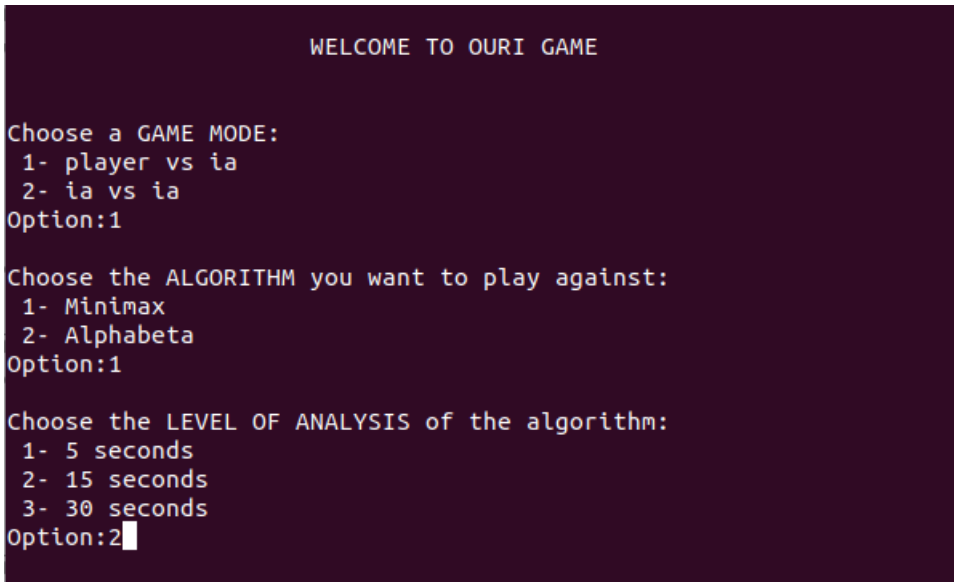

Figura 2: Menu para jogador contra algoritmo

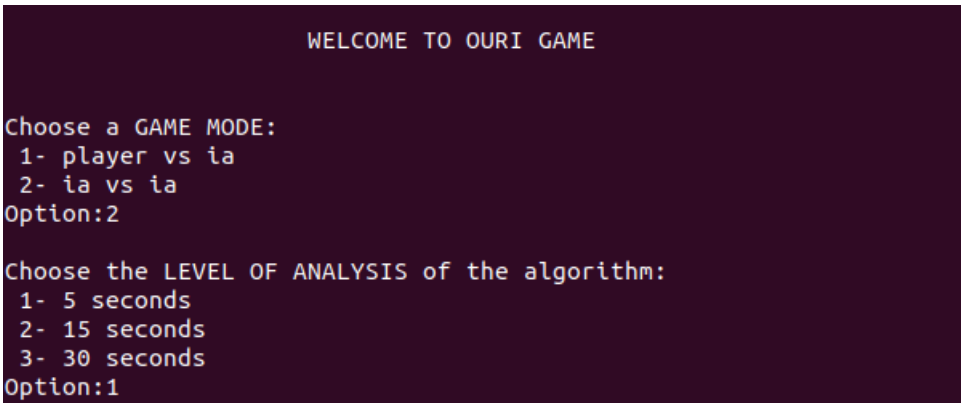

Figura 3: Menu para minimax contra alfa-beta

6 Bibliografia

- Regras do Jogo Ouri
  http://www.mat.uc.pt/ mat0821/regras%20ouri.pdf
  24 Junho, 15:42

- Minimax with Alpha-Beta Pruning in Python
  https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/
  24 Junho, 16:24

- Algoritmo Minimax - Introduçaõ à Inteligência Artificial
  https://www.organicadigital.com/blog/algoritmo-minimax-introducao-a-inteligencia-artificial/
  24 Junho, 21:06