Eximo

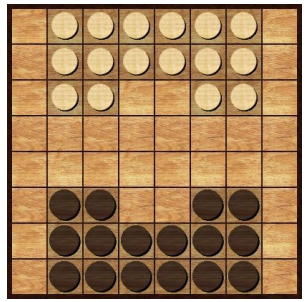
Gustavo Torres up201706473

Joaquim Rodrigues up201704844

Miguel Rosa up201706956

Especificação

- O trabalho a realizar consiste num jogo de 2 jogadores, similar ao jogo “Damas”, e procurar as melhores jogadas
- Objetivo:
  - Capturar todas as peças do adversário ou o adversário não tem mais jogadas possíveis.
- Regras:
  - As peças começam em posições predefinidas como mostra a imagem
  - Ordinary Move - movimento para a frente ou avançar diagonalmente para a frente
  - Jumping Move - saltar por cima de uma peça do jogador para a frente ou diagonalmente no mesmo sentido
  - Capture - saltar por cima de uma peça do adversário, capturando-a, para a frente ou avançar diagonalmente para a frente ou para os lados
  - Uma vez efetuado um Jumping Move ou Capture é necessário utilizar essa peça para fazer o mesmo movimento ate não haver jogadas possíveis, não necessariamente a jogada mais longa
  - Drop – se uma peça chegar à última fila do tabuleiro, essa peça é removida do tabuleiro e são adicionadas duas peças em posições na *drop zone* (duas primeiras filas do tabuleiro, exceto os 4 quadrados nas laterais)



Formulação do problema

- Para representação do problema recorremos essencialmente à representação do tabuleiro e um conjunto de *flags* capazes de representar o estado do jogo em qualquer momento. O tabuleiro de 8x8 é representado com recurso a 2 números de 64bits, um número para cada jogador, e cada bit indica se naquela posição se encontra uma peça no tabuleiro. O objetivo é chegar a um tabuleiro que represente uma vitória para o jogador.
- Para tal, na pesquisa da próxima jogada vai ser usado o algoritmo **minimax**, com recurso a cortes *alfa-beta*, de forma encontrar a melhor jogada de acordo com a heurística de avaliação do tabuleiro.
- A avaliação do tabuleiro vai consistir em diversos parâmetros, desde a existência de um *gameover*, numero de peças no tabuleiro de cada lado, disposição das peças no tabuleiro e peças que se encontrem em posição desfavorável (podem ser capturadas). Outras versões serão desenvolvidas e testadas a sua eficácia.
- O estado inicial do jogo segue as regras explicadas previamente.

Trabalho de implementação

- No seguimento do que foi descrito nos slides anteriores, já foi iniciada a implementação do projeto.
- A linguagem de programação escolhida foi C++, e a representação visual será feita na consola. O ambiente de desenvolvimento escolhido é o Visual Studio Code.
- No Checkpoint 1:
  - O jogo e as suas regras encontram se implementadas, sendo já possível jogar Humano *vs* Humano e Humano vs Máquina;
  - É possível obter todas as jogadas possíveis para um dado tabuleiro para um dado jogador;
  - Existe uma heurística de avaliação do tabuleiro, no entanto ainda vai ser melhorada;
  - Algoritmo minimax implementado;
  - Interface e visualização do tabuleiro já trabalhada;
  - Implementação de Iterative Deepening e Transposition Table.
- Na entrega final (para além dos pontos já cobertos no Checkpoint 1):
  - Todos os modos de jogo estão implementados (Máquina vs Máquina a adicionar aos outros 2 previamente implementados);
  - Existem três heurísticas de avaliação do tabuleiro;
  - Todos os algoritmos implementados (Minimax, Iterative Deepening e Transposition Table);

Abordagem

Para a avaliação do tabuleiro, recorremos a três heurísticas diferentes:

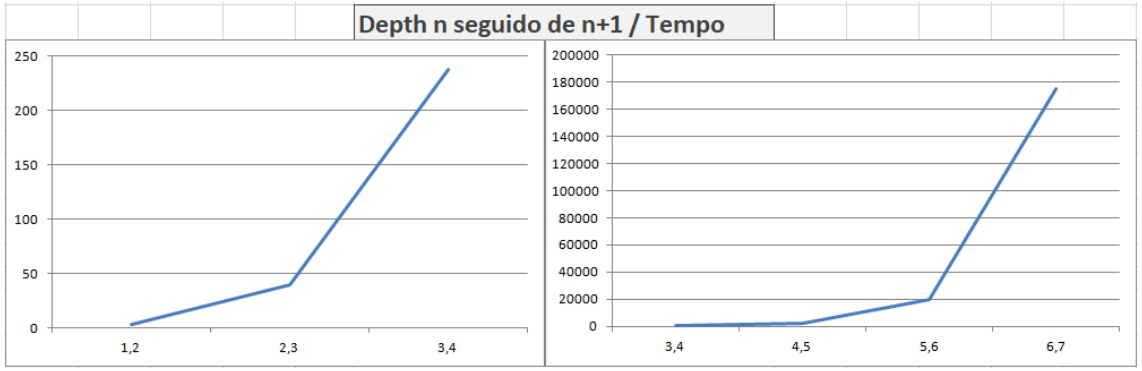
- 1. Avaliação do número de peças de cada jogador, exclusivamente. Trata-se da heurística mais greedy das implementadas;
- 2. Para além da heurística utilizada no ponto 1, tem-se em conta também o desenvolvimento do board nas pontas e das peças que se encontram na penúltima fila do adversário, de forma a serem convertidas em duas peças na parte do board do jogador. Trata-se da heurística mais agressiva;
- 3. Heurística no ponto 2, à qual se junta a consideração do desenvolvimento de peças no meio do board e das peças na primeira linha para defender o seu lado. Esta heurística é a que tem mais atenção à formação das peças no board e ao desenvolvimento geral no jogo.

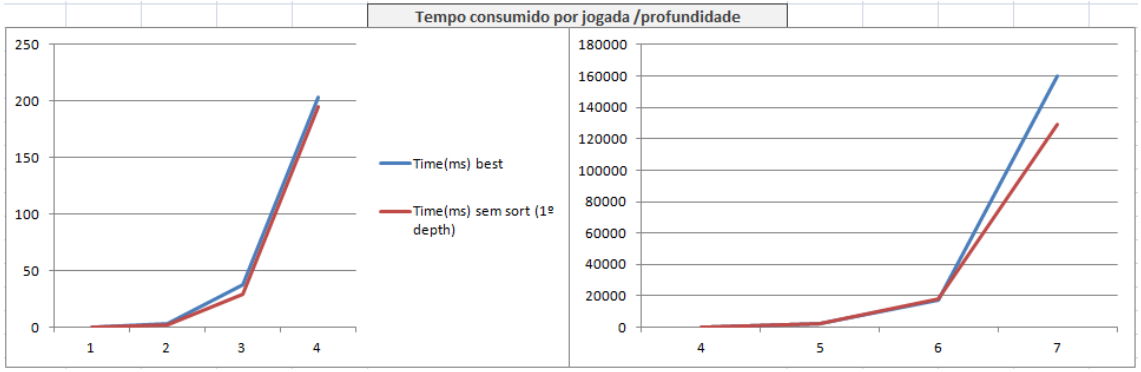
É de notar que para cada um dos parâmetros avaliados em cada heurística são atribuídos pesos diferentes, de acordo com o seu valor tendo em conta a vitória como objetivo final.

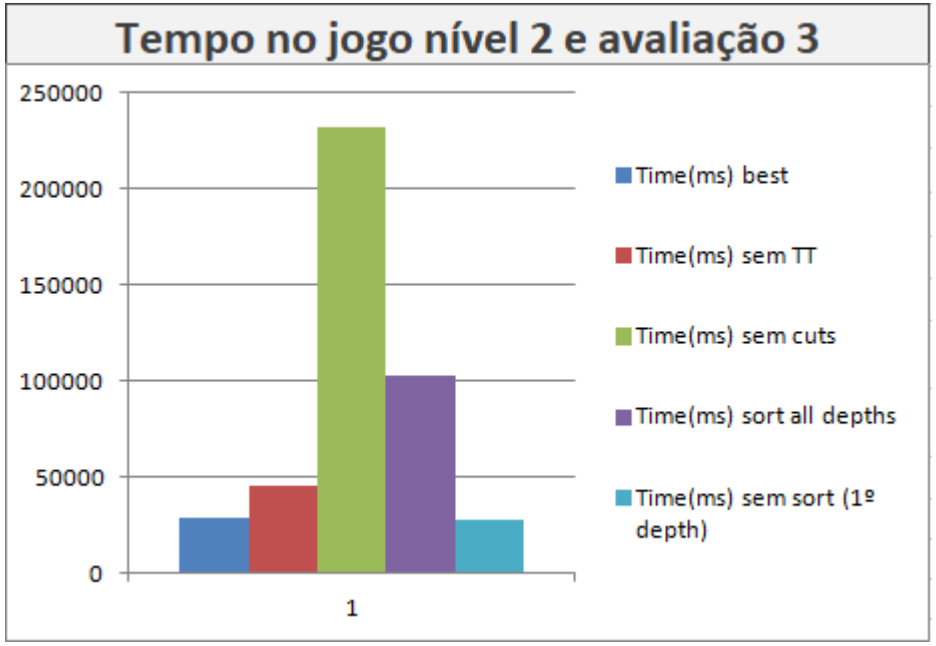
Algoritmos implementados

- Para a elaboração deste projeto era crucial a implementação do algoritmo **minimax**, com cortes alfa-beta, de forma a obter as melhores jogadas possíveis de acordo com a heurística em utilização. Os cortes alfa-beta são fundamentais para a redução do número de nodes analisados, uma vez que "corta" os nós que não resultarão numa alternativa válida para melhor jogada no estado atingido;
- Outro algoritmo implementado foi a pesquisa **Iterative Deepening**, que neste caso utilizará o algoritmo minimax para pesquisar a uma maior maior profundidade no caso de não ter consumido todo o tempo ou atingido a profundidade máxima, usando muito menos memória, sendo portanto um algoritmo que melhora a eficiência do programa e também a sua tomada de decisões;
- Implementou-se também uma Transposition Table, que serviu para reduzir o custo para o programa, uma vez que posições/ tabuleiros anteriormente calculados são guardados, evitando-se o seu cálculo novamente

Resultados experimentais







Resultados experimentais

	Player 1		Número de Movimentos								
Player 2	LEVEL		1			2			3		
		EVAL	1	2	3	1	2	3	1	2	3
1		1	68	214	168	74	64	61	64	90	68
		2	71	75	94	87	78	85	88	57	61
		3	63	141	449	242	112	95	139	84	80
2		1	70	63	68	65	131	67	60	66	67
		2	63	69	71	111	83	116	101	60	108
		3	70	65	87	142	102	104	98	99	80
3		1	62	107	78	66	79	137	106	141	98
		2	66	69	78	88	59	140	92	106	63
		3	56	104	99	113	109	106	63	112	99

	Player 1		Avaliação do tabuleiro								
Player 2	LEVEL		1			2			3		
		EVAL	1	2	3	1	2	3	1	2	3
1		1	21.5	-10.5	-16.3	-28.2	-24.2	-26.8	-21.8	-23.8	-27.8
		2	25	-18.7	-28.27	-28.1	-14.35	-19.9	-16.4	-14.8	-30.9
		3	21.5	28.5	21.1	-16.73	-22.1	-14.85	-24	-26.3	-16.8
2		1	16.6	26.3	14.5	-14.5	14.3	-18.5	-16.2	-14.8	-14.1
		2	29.8	19.9	23.6	-18.5	19.5	6.25	4.5	-19.2	-14.1
		3	36.5	27.5	10.8	-7.1	22.8	4.9	16.3	-13	-15.15
3		1	21.97	35.8	27.8	18.8	-23	-9.8	5.7	-7.9	-18.75
		2	29.2	20.2	11.4	16.425	17.8	-15.5	20.75	-8.54	16.075
		3	19.35	5.425	17	8.65	7.925	17.5	18.75	11.45	08.01

Conclusão

Através da realização do projeto e da observação dos resultados experimentais (ver tópico anterior) obtidos, pode concluir-se que os algoritmos melhoram a eficiência temporal do programa até um certo limite. Observando o gráfico de tempo consumido por jogada em função da profundidade, verificamos dois dados importantes: o sort não diminui, efetivamente o tempo despendido numa jogada, podendo isto ser explicado por este ser usado para a ordenação de tabuleiros e não na organização aquando da geração de jogadas e também porque para efeitos de teste foi usada a jogada inicial que não influencia a Transposition Table suficientemente nem tem informação útil para se notar o seu efeito, e dá-se também o aumento de tempo aumentando a profundidade de pesquisa, como de facto seria de esperar (algo confirmado no gráfico depth n seguida de n+1 por tempo).

Para além disso, no gráfico de barras do tempo médio no nível 2 e avaliação 3 (a mais completa), verifica-se a diferença significativa do uso dos cortes alfa-beta no tempo de cada jogada. Estes dados também eram expectáveis, dada a função desses cortes, eliminando a pesquisa em nós da árvore que não produziam resultados favoráveis.

Nos últimos dois quadros verificamos, salvo poucas exceções, que aumentando o nível e a avaliação utilizados por um jogador virtual, aumenta-se também a probabilidade desse jogador virtual ganhar. Quando ambos os jogadores virtuais usam a mesma avaliação do tabuleiro e o mesmo nível denota-se que os jogos costumam ser mais longos, sendo a avaliação aproximada a 0.

Neste trabalho conseguimos perceber como encontrar jogadas para certos problemas de informação perfeita, através do algoritmo minimax. No entanto, devido à complexidade de jogadas do jogo não nos foi possível obter a melhor jogada, forçando-nos a restringir a profundidade máxima do nosso algoritmo e a obter uma aproximação da melhor jogada possível para um determinado momento. Conseguimos otimizar o tempo de execução e deste modo ir a uma profundidade superior com recurso a diversas técnicas de otimização.

Bibliografia

- https://www.boardgamegeek.com/boardgame/137916/eximo;
- https://www.youtube.com/watch?v=l-hh51ncgDI;
- https://www.chessprogramming.org/Iterative_Deepening;
- https://en.wikipedia.org/wiki/Transposition_table;
- https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning;