




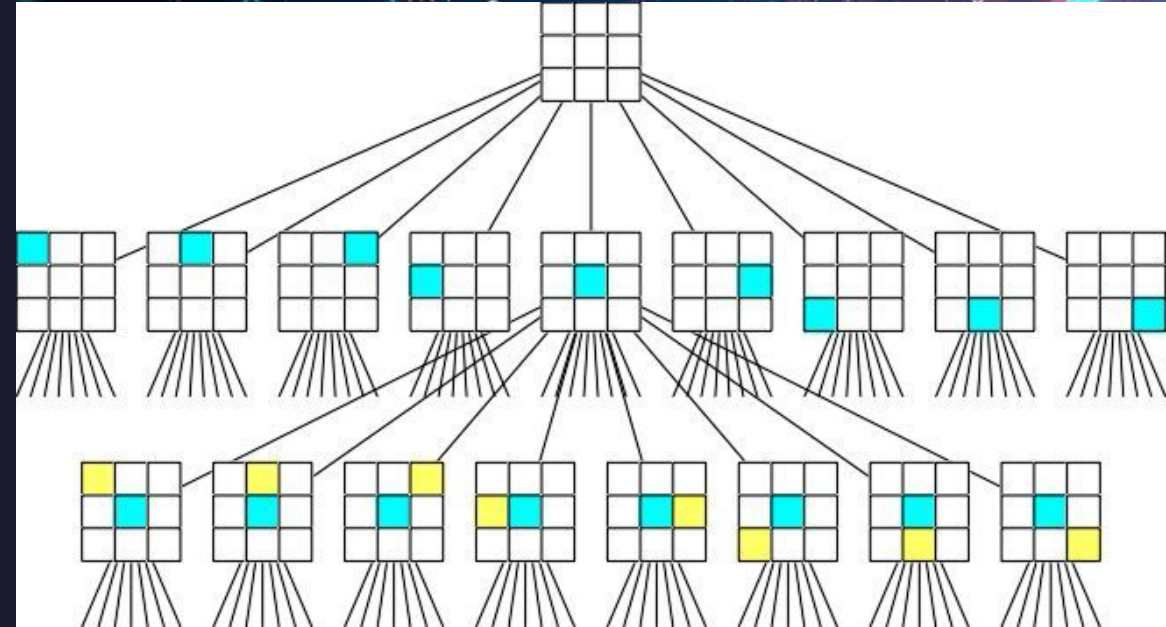
Jogo da velhAI

Inteligência Artificial criada para jogar o "jogo da velha". Foi utilizado um algoritmo de busca competitiva, mais detalhadamente o **algoritmo MiniMax** de teoria dos jogos.



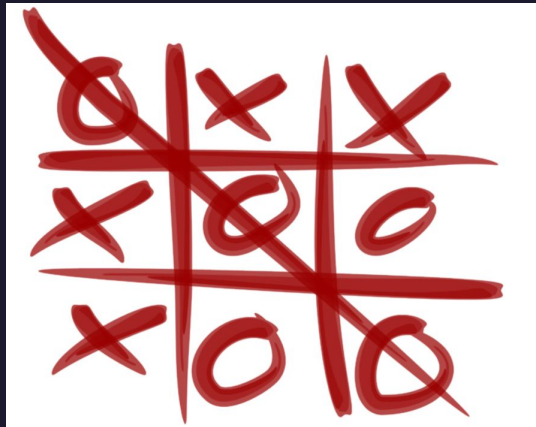
Introdução

- A ideia central é modelar a competição estratégica como um processo de tomada de decisão baseado em matemática. O algoritmo Minimax, derivado desses princípios, se baseia na premissa de que dois jogadores, cada um buscando maximizar seus próprios ganhos, podem ser representados através de uma árvore de decisão



Algoritmo Minimax

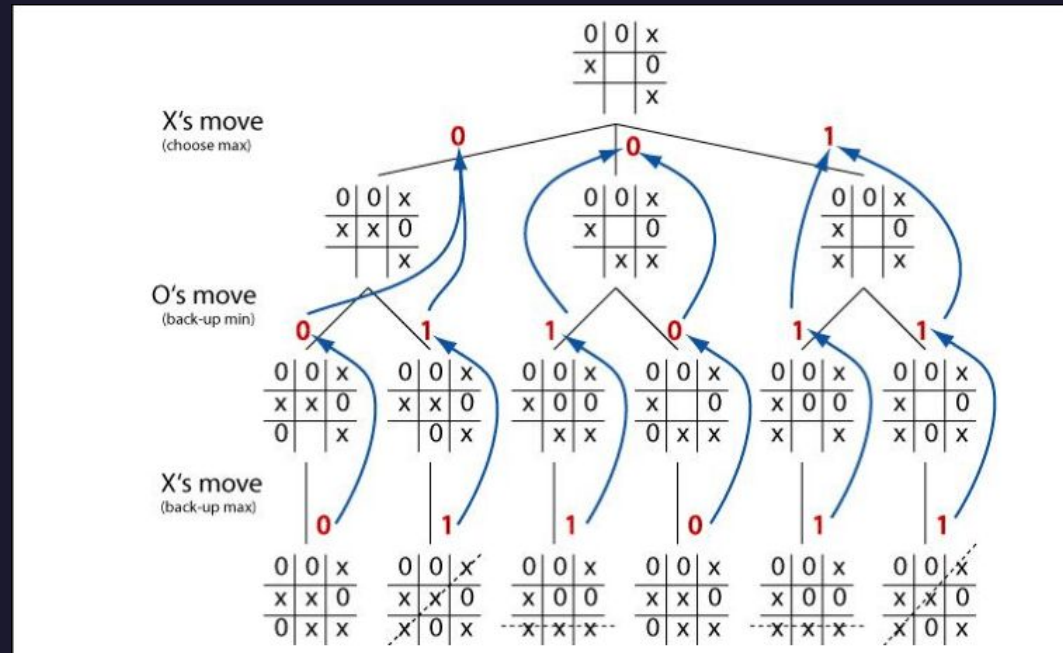
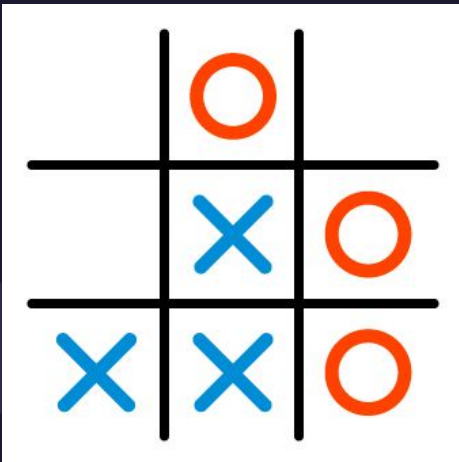
O algoritmo Minimax é amplamente utilizado em jogos de estratégia para tomar decisões em situações de adversidade. Ele **analisa todas as jogadas possíveis**, alternando entre maximizar e minimizar os resultados, com o objetivo de encontrar a melhor jogada para um jogador, assumindo que o adversário também joga de forma ideal.



Árvore de Decisão

Nesta parte do algoritmo, gravaremos o que chamamos de “estado” que são cenários possíveis do jogo. Faremos isso gerando uma árvore de decisão, a partir do nosso estado atual.

Ela representa todas as possíveis jogadas e estados do jogo, permitindo que o algoritmo avalie as diferentes opções de maneira sistemática.



Construção da Árvore

No início, a raiz da árvore representa o estado atual do jogo. Para cada possível jogada disponível, cria-se um novo nó na árvore, representando o estado resultante após aquela jogada. Isso é repetido recursivamente para todos os estados subsequentes até atingir um estado final, como uma vitória, empate ou derrota.



Avaliação dos Nós

No início, a raiz da árvore representa o estado atual do jogo. Para cada possível jogada disponível, cria-se um novo nó na árvore, representando o estado resultante após aquela jogada. Isso é repetido recursivamente para todos os estados subsequentes até atingir um estado final, como uma vitória, empate ou derrota.

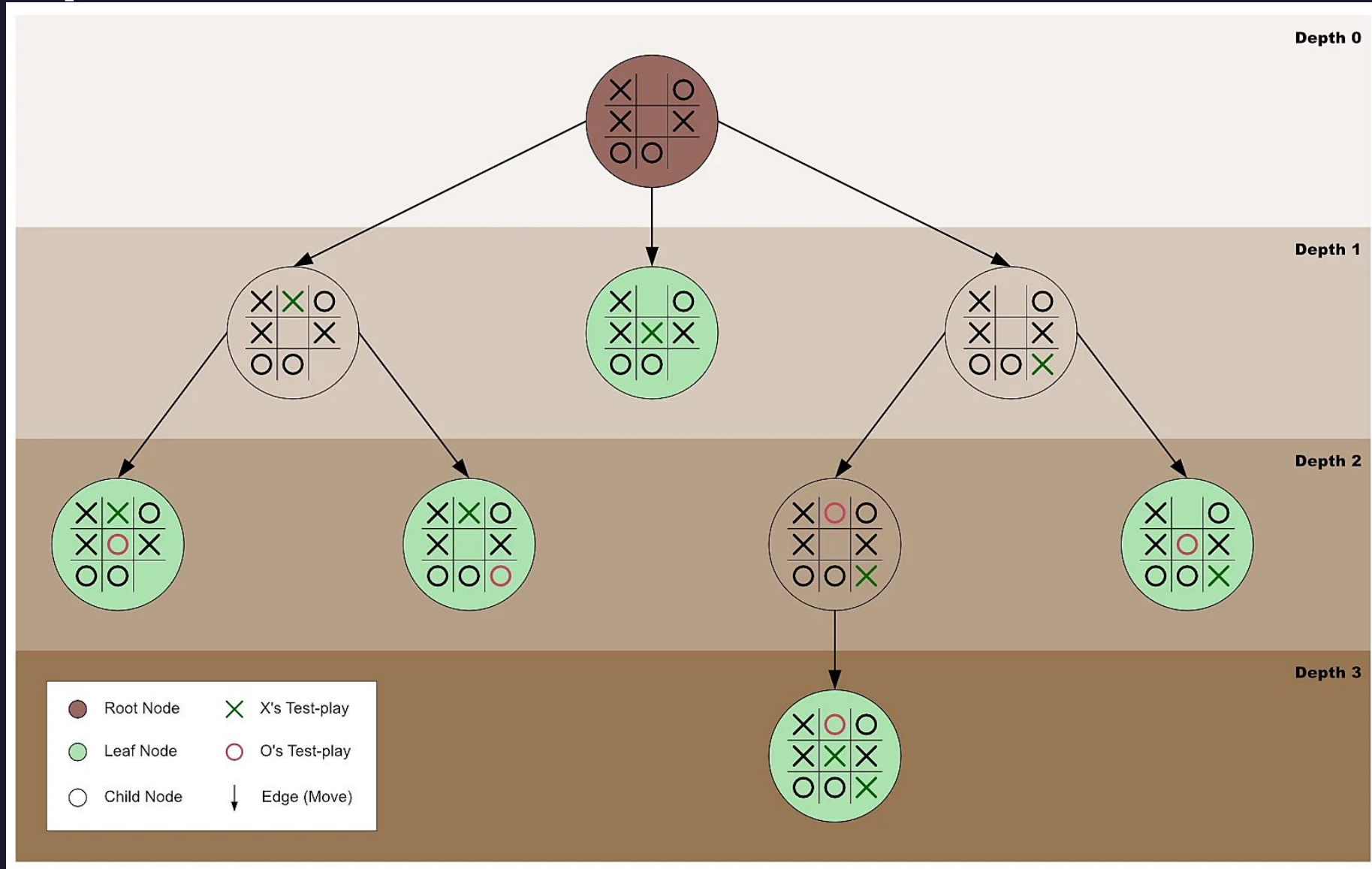


Seleção da Melhor Jogada:

Após a construção da árvore e a atribuição de valores a todos os nós, a IA escolhe a jogada que leva ao estado com o valor mais alto na raiz da árvore. Essa jogada é considerada a melhor estratégia a ser seguida, assumindo que o adversário também jogará de forma ideal.



Exemplo



O min e o max

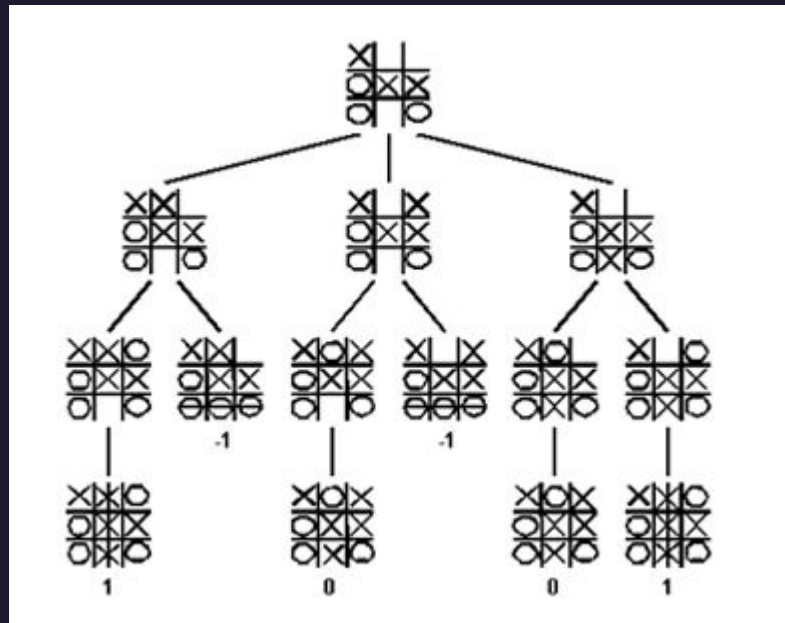
Baseia-se na ideia de que, em um jogo estratégico, dois jogadores estão competindo entre si, e um jogador deseja maximizar seus ganhos, enquanto o outro deseja minimizar suas perdas.

- **Nós de Maximização:** Quando o algoritmo Minimax se depara com um nó onde é a vez da IA jogar, ele busca maximizar o valor deste nó. Isso significa que a IA escolhe a jogada que levará a um estado com o maior valor possível. Em outras palavras, ela tenta encontrar a jogada que maximiza seus próprios ganhos potenciais.
- **Nós de Minimização:** Por outro lado, quando o algoritmo chega a um nó onde é a vez do adversário jogar, ele busca minimizar o valor deste nó. Isso reflete a suposição de que o adversário fará a jogada que levará a um estado com o valor mais baixo possível. O objetivo aqui é evitar que o adversário alcance uma posição favorável.



O min e o max

Em resumo, o algoritmo Minimax usa valores simplificados a nível de implementação como **-1 para derrota, 0 para empate e 1 para vitória**, para atribuir pontuações a cada estado do jogo e decide suas jogadas com base em como cada jogada afetará essas pontuações, buscando maximizar suas próprias chances de vitória ou minimizar as chances do adversário.





Como otimizar?

Poda Alfa-Beta

- Visto que este algoritmo é um algoritmo de busca, ou seja buscamos a melhor opção e após fazer todo o backtracking, iremos selecionar a melhor opção. É possível pensarmos que podemos fazer a famosa poda(pruning the search).

Poda Alfa-Beta

- Visto que este algoritmo é um algoritmo de busca, ou seja buscamos a melhor opção e após fazer todo o backtracking, iremos selecionar a melhor opção. É possível pensarmos que podemos fazer a famosa poda(pruning the search).
- E os estudos de minimax sugerem que isso é possível!

Poda Alfa-Beta

- Visto que este algoritmo é um algoritmo de busca, ou seja buscamos a melhor opção e após fazer todo o backtracking, iremos selecionar a melhor opção. É possível pensarmos que podemos fazer a famosa poda(pruning the search).
- E os estudos de minimax sugerem que isso é possível!
- Essa técnica permite reduzir a quantidade de nós explorados na árvore de decisão, economizando tempo de processamento.

Poda Alfa-Beta

- Visto que este algoritmo é um algoritmo de busca, ou seja buscamos a melhor opção e após fazer todo o backtracking, iremos selecionar a melhor opção. É possível pensarmos que podemos fazer a famosa poda(pruning the search).
- E os estudos de minimax sugerem que isso é possível!
- Essa técnica permite reduzir a quantidade de nós explorados na árvore de decisão, economizando tempo de processamento.
- Ela permite otimizarmos o algoritmo de forma a ignorar a avaliação de certos ramos(ou melhor estados) da nossa árvore de decisão

Implementação

- Quando exploramos os nós da árvore de decisão, manteremos duas variáveis: **alpha** e **beta**.



Implementação

- Quando exploramos os nós da árvore de decisão, teremos duas variáveis: **alpha** e **beta**.
- Alpha representa a pontuação mínima garantida que o jogador Max (a IA) já alcançou até aquele ponto na árvore



Implementação

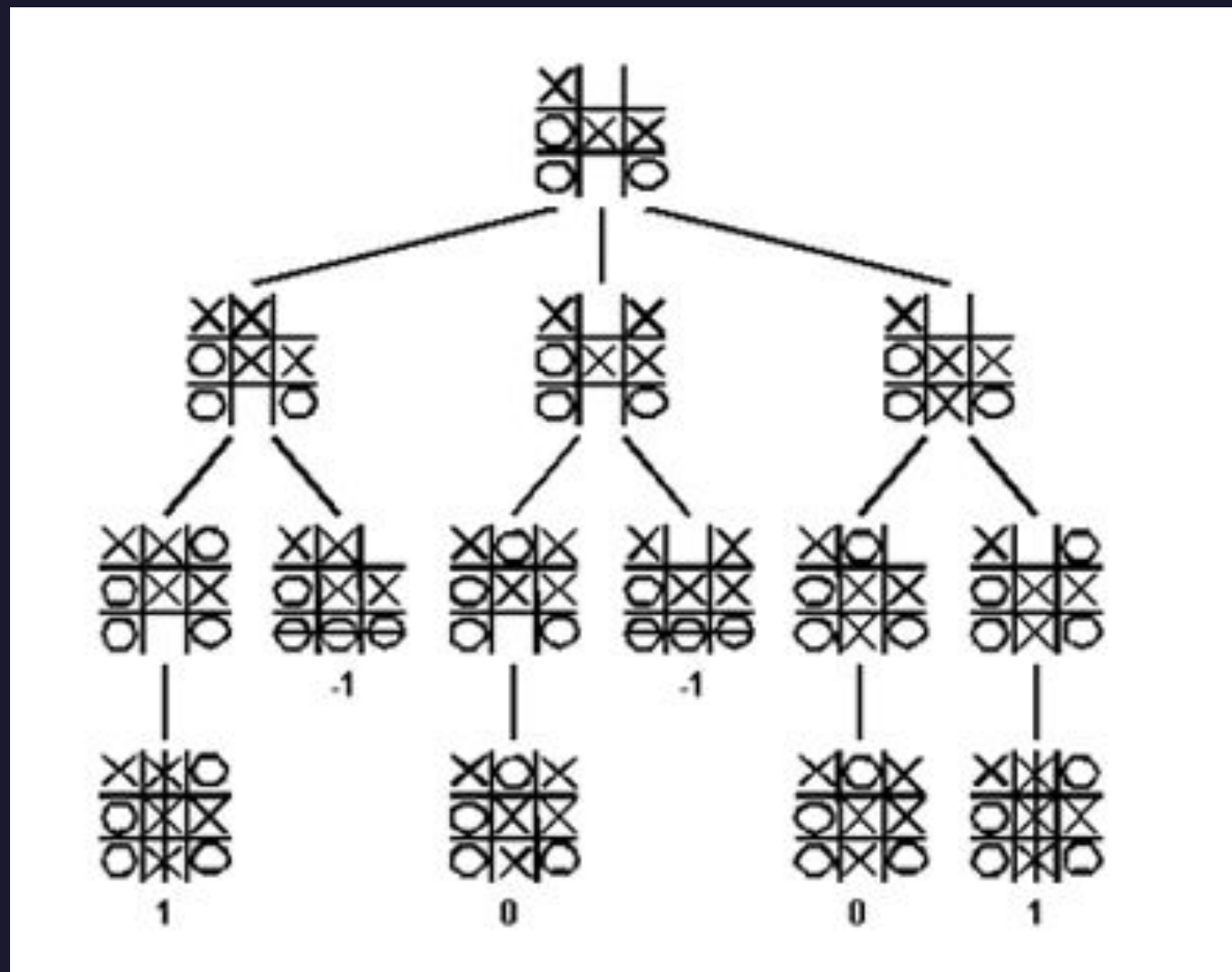
- Quando exploramos os nós da árvore de decisão, teremos duas variáveis: **alpha** e **beta**.
- Alpha representa a pontuação mínima garantida que o jogador Max (a IA) já alcançou até aquele ponto na árvore
- Beta representa a pontuação máxima garantida que o jogador Min (o adversário) já alcançou até aquele ponto na árvore.



Implementação

- Quando exploramos os nós da árvore de decisão, teremos duas variáveis: **alpha** e **beta**.
- Alpha representa a pontuação mínima garantida que o jogador Max (a IA) já alcançou até aquele ponto na árvore
- Beta representa a pontuação máxima garantida que o jogador Min (o adversário) já alcançou até aquele ponto na árvore.
- A ideia é que, se você encontrar um nó onde o valor seja maior que beta (para jogador Max) ou menor que alpha (para jogador Min), você pode cortar a exploração dos ramos restantes, pois eles não afetarão a decisão final.

Implementação



Obrigado!



Projeto: **Jogo da velhAI**

Matheus Costa Monteiro

Carlos Humberto Martins