# How I defeated baby agent

Minimax Algorithm

The initial approach to building the AI involved implementing the Minimax algorithm. This decision-making algorithm works by minimizing the possible loss in a worst-case scenario, enabling the AI to make optimal moves.

Evaluation Function

For the evaluation of board states, I calculated the maximum number of connected player cells in a row. This heuristic allowed the AI to assess the potential strength of a position based on immediate threats and opportunities, helping to guide decision-making.

Depth Limitations

Initially, I was only able to set the maximum search depth to 4 as any more would take too long. However, this limitation hindered the AI’s effectiveness, resulting in losses against the baby agent. The shallow depth restricted the AI's ability to foresee future moves and potential strategies.

Alpha-Beta Pruning

To enhance the performance of the Minimax algorithm, I incorporated Alpha-Beta pruning. This optimization technique significantly reduced the number of nodes evaluated in the search tree, allowing me to increase the maximum search depth to 6. As a result, the AI was able to defeat the baby agent, demonstrating improved strategic planning.

Performance Challenges

Despite the advancements achieved through Alpha-Beta pruning, the AI's response time became a concern. The increased depth, while enhancing decision-making capability, also led to longer computation times, making it difficult to maintain a reasonable pace during gameplay.

Memoization

To address the timing issues, I implemented memoization. This technique involved storing previously computed values of board states, effectively eliminating redundant calculations during the search process. By caching results, the AI could retrieve evaluations of previously encountered positions quickly, leading to a significant reduction in response time.

Results

With the combination of Alpha-Beta pruning and memoization, the AI was able to beat the baby agent within the time limits set for gameplay. The improvements allowed the AI not only to make strategic decisions more effectively but also to do so efficiently.

# Improvements to defeat base agent and improve score

Connect 4 8-ply database

I used the Connect 4 8-ply database to improve the AI's performance. By hashing the precomputed results in the database, I was able to compress the data such that it can be integrated into the AI seamlessly. With that, the AI can quickly seek up game conditions that are known to result in winning or losing outcomes. By hashing each state that it encounters in the minimax algorithm, this allows the bot to make more informed moves that lead it to the winning states as defined in the database.