HW3: Connect 4

My code for an AI connect 4 agent uses a basic minimax algorithm to generate the best move to make at any time. The code constructs a game tree using all possible moves that can be made, up to a given depth. It then traverses the tree up to this depth and simulates the board at each point with that move made. For each possible move it makes, it evaluates the board at that position and assigns it a numerical score, ranging between -∞ and ∞. A score of -∞ means that the blue player (the AI agent in this scenario) has an absolute advantage and has won the game by placing four chips in a row. Conversely, a score of ∞ means that the red player (the ‘human’ in this case) has won. A score of zero means that both sides are doing equally well.

The code takes advantage of processing power to quickly evaluate many states in the game tree to determine which gives the best score for that player. For instance, if the program sees that a few levels down a certain move can result in a win, then it will assign a high (or low, depending on who is minimising and who is maximising) score to that subtree and choose the move that leads to that scenario.

The algorithm also utilises alpha-beta pruning to keep computation time reasonable, by discarding subtrees that have no way of yielding a better result than we can already achieve.

In my testing, I found that running the AI agent against a random agent 100 times usually resulted in between 95-99 wins for blue (the AI agent), with the rest being won by red (the agent playing randomly), for a search depth of 3.