Vivian Gunawan

Programming Assignment 2

Created a function called scoresFor within the AIPlayer class that evaluates the board. This function works by counting the number of “lines” on the board in different orientations(horizontal, vertical, diagonals). The lines are classified by the number of running pieces (or connected pieces) and if they are blocked by an opponent checker or not. Then different scores are assigned accordingly by the calculate function. The way the scores are assigned is pretty straight forward if you look at the code. I set the ‘X’ checker to be the minimizing player while the ‘O’ checker to be the maximizing player. There are a lot of helper functions written to help me detect the lines. In terms of the terminology mentioned in the assignment, the above is the evaluation of “the heuristic value of a node”. The node represents the current state of the board in a given turn, where children of the node represent the state of the board during the next turn. A “terminal node” is equivalent to a winning state for one of the checkers or if there is a tie filling the whole board.

The minimax algorithm I implemented is of a shallow depth since deeper trees even with alpha-beta pruning exceeds the computational time of 5s. In the process of solving this programming assignment, I tried various methods, such as limiting the next possible moves to only cells near placed checkers when generating children of the nodes, I also tried increasing the depth. However, when competing the AIs against each other what I found work best given the limitation is a shallow tree that simply tries all the possible moves on the board. The AI I created performs better than just randomly selecting cells on the board and manage to win, however it fails in blocking the opponent in cases like XX_XX where the opponent’s final move is in between pieces. I think this is more likely a fault in the evaluation function not being thorough and can be improved upon given more time.