Reilly McBride
Period 5
February 19, 2018

Othello Improvements Made:

I added a History Table to my AI, which saves moves with assigned weights based on their depth. The weighting of a move is changed or increased when that move is chosen as the "best" move at the end of a recursive call. When considering new moves, they are sorted based on their history table scores. Thus, moves that were seen before--deeper in the tree--have a higher score. This is based on the idea that the deeper a move is in the tree, the more reliable its score is. It's more likely that a given move, if it was the best deep in the tree, will still be the best if its seen again—even at another depth. The deeper the node is, the less options for moves occur after it, the AI is more likely to be correct when it selects it (seeing as it cannot fully play out the game every time). The History Table, which is a dictionary created outside the alpha-beta method, is initialized with every index where someone can play on the board, and the indices are updated when the move is seen again. A new weighting is simply added to the previous weighting. This strategy came from "The History Heuristic and Alpha-Beta Search Enhancements in Practice", by Jonathan Schaeffer.

To account for potentially very high weightings if my code reached a low depth, I increased the factor I multiplied final board scores by to 1 million (instead of the initial 1000). Thus, there would be no chance of a high weighted move found deep in the tree overpowering a certain win.

From my own observation, I changed the weighting matrix to weight the two squares next to the corners (e.g., (1,2) and (2, 1)) as 20 instead of -20. In almost every game I played, it seemed that my code was choosing worse moves and losing because it was ignoring valuable close-to-corner moves, as their weighting wasn't high enough for it to view them as beneficial moves. I saw an immediate improvement after implementing this new weighting matrix.

Reilly McBride
Period 5
February 19, 2018