This project is an exploration of writing an AI for a zero-sum two-player full-information game. In this case, Reversi (or Othello; same game). Reversi’s a game that I’ve always been interested in, in spite of—or perhaps because of--being such a terrible player of it.

The goal of the game is to have the most tiles at the end when there are no more moves possible. Every turn you add a tile of your color to the board and flip any enemy tiles to your color that you have just surrounded. You can only place your tile in a position that would cause at least one enemy tile to flip. You can read the rules in more detail here.

So you can of course easily play human vs human just by clicking on Start Game and then making your move. The interesting part though is playing against the AI or letting the AI play itself. Clicking the *Configure AI* button lets you choose between a number of AIs. The other options are only relevant to particular AIs.

* AI Random Chooses randomly based on the moves available.
* AI Maximize Chooses the play that flips the most tiles.
* AI Minimize Chooses the play that flips the fewest tiles.
* AI First Chooses the first valid move in reading order from the top left.
* Best Leaf Choose the play that leads to the best possible outcome after Max Depth possible moves into the future.
* NegaMax Choose the play that leads to the least-bad worst-case outcome after Max Depth possible moves into the future. AB Pruning can be applied here.
* ProofNumber Choose the play that minimizes the number of plays your opponent can do, looking as far into the future as it can during Max Time seconds.

Best Leaf, NegaMax, and ProofNumber are the most interesting ones. They each build a game tree: a tree of possible moves looking into the future. On each turn there’s an average of 8.5 possible moves; therefore, looking n moves into the future requires examining about 8.5 ^ n board arrangements. 10 moves into the future is about 2 billion and 12 moves into the future is about 140 billion. It’s generally impractical to look far enough into the future to see the end of the game.

For NegaMax (and only NegaMax) I implemented a/b pruning as an optional optimization. The pruning is as the name implies: it allows me to cut branches off the game tree when it knows that the given branch is not going to contain the best solution. This allows for a huge reduction in the number of game states examined and lets the AI look much farther into the future much faster.

We still can’t reliably see to the end of the game though, so the other half of the battle is figuring out how valuable a particular game board arrangement is. I tried a few things but settled on valuing tiles that can no longer be flipped (like the corner) above all others and secondarily just value more tiles over fewer. This consistently worked the best through trials vs a random AI. Best Leaf and NegaMax use this but the ProofNumber algorithm does something different.

ProofNumber builds an what’s called an AND-OR tree (<https://en.wikipedia.org/wiki/And–or_tree>) as its game tree. The AI continuously expands the most easily proved branch of the tree, where proving a branch means you win. Your move options are considered ORs and the enemy’s are ANDs, and the provability of a branch is based on counting up the *AND*s and choosing the *OR* that leads to the fewest *AND*s. The outcome is that the computer tries to be controlling and minimize your move options, while steering towards the shortest game possible.

AI in games has been an interesting experiment so far. As an unrelated bonus, I am now a much better player of Reversi. Next steps are to implement a learning algorithm that would use a neural net and learn how to play the game.

Pressing *Statistics* after a match vs NegaMax