Benjamin Jacob

**HW3 docs:**

Using the minimax search algorithm, I was able to create an algorithm that beat the average player at Connect 4 (or at least against me :) ).

Alpha Beta pruning was used to make the algorithm a bit faster, but the more layers deep you go, it still gets much slower, as there is more and more that has to be done at each step down the tree.

My heuristic was made using 3 separate factors:

1. The obvious one: having multiple in a row – if there are 2, 3 in a row: with nothing in the other spots needed to get to 4 (ex: **1-0-1-0** and not **1-2-1-0**), then a particular board would be worth more. If there were 4 in a row, then it would be pretty fantastic, and that would be given a lot of points.
2. Double traps: if there were multiple ways to get to 4 in a row – this would be a case where there are 3 in a row (with an empty space next to it) in multiple places on the board, so you automatically get to 4 in a row on the next more, no matter what your opponent chooses.
3. Opponent’s ability to win: this part is more or less the opposite of steps 1 & 2: make sure that your opponent doesn’t have a good chance of winning – however much a board would be boosted for being good for our algorithm, it should be almost be equally bad if the opponent is given could positioning (however, not the same amount, as then our algorithm may be playing way too much defense.

Some other notes:

* 2 things are checked before starting an iteration of the minimax algorithm:

1. If I can just win: if I can win by putting a piece down in one of the columns, then there’s no point in entering into the algorithm.
2. If I have to block my opponent: if my opponent has 3 in a row and can win next turn, I have no other choice but to block(assuming I did step ‘a’ and checked if I can win or not)