*Project overview:*

My project is Connect 4. It uses standard Connect 4 rules. The human player goes first, followed by the computer player. The human player uses “X”s while the computer player uses “O”s.

To run the program, compile and load the file into Allegro, and start the game by calling the function ***(play)***. Then enter which column you wish to play when prompted. The game will end when either player has 4 in a row, or the board is full.

Note that there are a ton of compilation warnings that pop up when you compile it. I don’t know why they all show up, I don’t know enough about either Allegro or Lisp. But despite the warnings the program runs just fine.

*AI Strategy Algorithm:*

The AI will look at each potential move, and assign it a score based on what combinations of consecutive pieces it will make for itself, or block the human player from making.

50000 to create a 4-in-a-row

10000 to block a player from making a 4-in-a-row

5000 to create an open-ended 3-in-a-row (a 3-in-a-row with a playable space on either side of it)

1000 to block a player from making an open-ended 3-in-a-row

100 for each 3-in-a-row created or blocked

10 for each open-ended 2-in-a-row created or blocked (a 2-in-a-row with a playable space on either side of it, and one or both of the sides has at least two playable spaces)

1 for each 2-in-a-row created or blocked

Note that for each combination, there must be enough spaces on either end for it to potentially be a 4-in-a-row, otherwise it is ignored. The scores are cumulative, so a move that will create two 2-in-a-rows and block one player 3-in-a-row will score 102 points.

After assigning each move a score, the AI will first look for winning moves, both for itself or to block the human player from making. If there aren’t any, before selecting a move the AI will “think ahead.” First it will look at the human player’s next move, removing from consideration any move that will allow the player to create a four-in-a-row or an open-ended 3-in-a-row on the following turn. If there are any remaining moves that give it an open-ended 3-in-a-row, it will play that.

If not, it will then think further ahead to its next turn. It will look at any moves that by playing that move, it sets up a potential 4-in-a-row or open-ended-3-in-a-row (ignoring moves removed from consideration in the previous step and moves that already create a 3-in-a-row—getting a four-in-a-row from these are obvious, so it looks to create more “sneaky” potential winning moves). If there are any such moves, then that score replaces the current score for that move. The AI will then play the move with the highest score; if multiple moves have the same high score, it will choose one randomly. To see the flow of the AI decision process, see the AI decision tree diagram below.

I think the program could possibly be set up to call the process of thinking ahead recursively and look ahead as many turns as you want, but I am sticking with 1 for this project. If I had more time I’d experiment with multiple moves ahead looked at, and letting the player choose an opponent difficulty, and using that to determine how many moves ahead the AI will look at.

Does any move give the computer 4IR?

Does any move block the player from getting 4IR next move?

Make this move

Can this move give the player a 4IR or OE3IR next move?

Change score to -1 (don’t play)

4IR = Four-in-a-row

OE3IR = Open-ended three-in-a-row

Change current score to this score

Will this move give the computer an OE3IR?

Evaluate the other moves

Will playing this set up a potential 4IR or OE3IR the following move?

Keep current score

After going through all moves, get move with best score