Introduction

We picked this project because we are interested in learning, coding, and AI. One question we were hoping to answer was: how would one go about creating an AI? Then how would one connect it back to Connect 4? Creating an AI connects back to material that we have seen in class through the class presentations of AGI and through that the problem of identity.

Program

We created an AI for Connect 4, along with a moderator program, which allows for playing against, or making another player play against, our AI. First, we created an empty six by seven grid, and a function to print it. We created horizontal, vertical, and diagonal checks, which will be elaborated on more later. After creating these checks, we created an input function, to allow testing these checks. This was all placed in a single file (pabwselConnect4Check.py). We moved the main gameplay to another file for extra clarity (pabwselConnect4Game.py). The main gameplay file is the moderator program.

Then, we finally got around to making the AI in a new file (pabwselConnect4AI.py). Through trial and error, we figured out where to use minimums or maximums (sometimes at the same time). We made our AI a single function that takes in the grid and the turn, and returns a number between zero and six.

Going back to the checks, we created a function that found the row index of the top piece in each column of the grid. This function was immensely important for the rest of the checks because it eliminated the need to search through the entire grid. Instead, our checks will start with the top values, and evaluate using them. Our vertical check started at the value listed in tops and counted down, and our horizontal check also only checked rows that contained values listed in tops. The values returned from both checks are sorted by column. The diagonal check is similar to the horizontal check and is also sorted by column.

The AI implements these checks by first checking the tops, then placing a piece in each column, one at a time. Each time, it checks for a win, and then places an opposing piece in each possible value, one at a time, to check for the other player's score. It then takes the maximum for the opposing piece, and places that in a list. The AI then places its piece in the minimum value in that list or the minimum value of its own list.

Results

Our program can check for diagonal, vertical, and horizonal win scenarios, to play against a human player, and to block a win of an opponent. The AI checks every column for possible ways that it and its opponent could win the game with in two turns. It can play against either itself or a human being depending on which game file is run. Our program blocks horizontal, vertical, and diagonal wins by assigning each column a value as it checks it.

Because we used a recursion technique to take player input, the user can break the program by repeatedly putting in random values that do not match the expected input causing the program to crash. However, this was sufficient and simple to use for our purposes.

Discussion

The program did satisfy our goals of creating an AI that can play connect-4 well by blocking its opponent from winning and actively attempting to win. The program relates to topics in class such as AGI and the problem of identity. It relates to AGI by being similar to Deep Blue and Watson where it has a specific function that it is extremely good at. However, it is nowhere near being AGI since it does not use machine learning in anyway and merely follows programming. It relates to the problem of identity through the AGI presentation. Its true identity is one python file that pulls from one python file and is run in another python program. The AI relies on the python file that contains the check functions and is called in the moderator program. One extension or improvement we would like to see in the future would be to have the program use machine learning to develop a winning strategy.