Assignment 2 - Connect 4 Game

Devasha Trivedi

October 23, 2022

1 Questions

1.1 What heuristic did you use? Why?

I decided to write my evaluation function in a way that the utility value is the result of win conditions of player two minus win conditions of player one. I did so keeping in mind that the player whose turn it is will be referred to as "player one" and the opponent will be "player two". In my code, it's presented in this way:

```
utility = self.chance_score(board, (self.player_number * 2) % 3) -
self.chance_score(board, self.player_number)
```

where the first term is taking the winning chances of player1 and the second term is taking the winning chances of player2. The utility returned helps my algorithm make a decision on what to do each turn based on how close the number is to zero.

1.2 Describe how your algorithm performs given different time constraints. How much of the tree can you explore given 5 seconds per turn? 10 seconds? 3 seconds?

My algorithm needs at least 2 seconds to be able to make a move without timing out. Within 3 seconds it's able to explore two depth levels of the tree and I capped it off at that for 5 and 10 seconds by introducint a depth limit, because exploring more than two depth levels made my algorithm more time-consuming per turn.

I did notice that between a random player and my algorithm, if I leave the time setting to the default 60 seconds, then the random player has a chance of winning. Any time constraint set $<= 40$ seconds ensured that my algorithm won.

For human vs. ai games, I didn't notice much of a change between how my algorithm behaved with different time limits.

1.3 Can you beat your algorithm ?

Initially, every single time. It was really easy to do if I manage to get into a situation where I have three chips together and an open slot on both sides (as depicted in Figure 1 below.). The only time the algorithm has beaten me is when I was more focused on blocking it than creating winning situations for myself.

Then, I got to try and improve my algorithm to look two moves ahead instead of one by working with the depth limit I had set and how I used it in my algorithm. Using that, I was able to create a better algorithm that caught the tricks I employ when I play this game against other humans. Figures 2 and 3 in the Images section show two different outcomes after this improvment was implemented, and in one of them my algorithm beat me (though I also attribute this to exhaustion from playing the game so much during this assignment).

1.4 If your algorithm plays itself, does the player that goes first do better or worse in general? Share some of the results.

Every time my algorithm played itself before the aforementioned depth limit improvement, player 2 won. A sample result is shown in Figure 4 below. It did a good job of blocking itself from winning but when it doesn't have a column to block, it tended to fill up the columns that still had space; which is not necessarily productive.

After the improvement, player 1 began to win every time. I'm not quite sure the reason for the sudden change, but it was certainly interesting to notice. Figure 5 shows the outcome after the depth limit change I made.

2 Figures

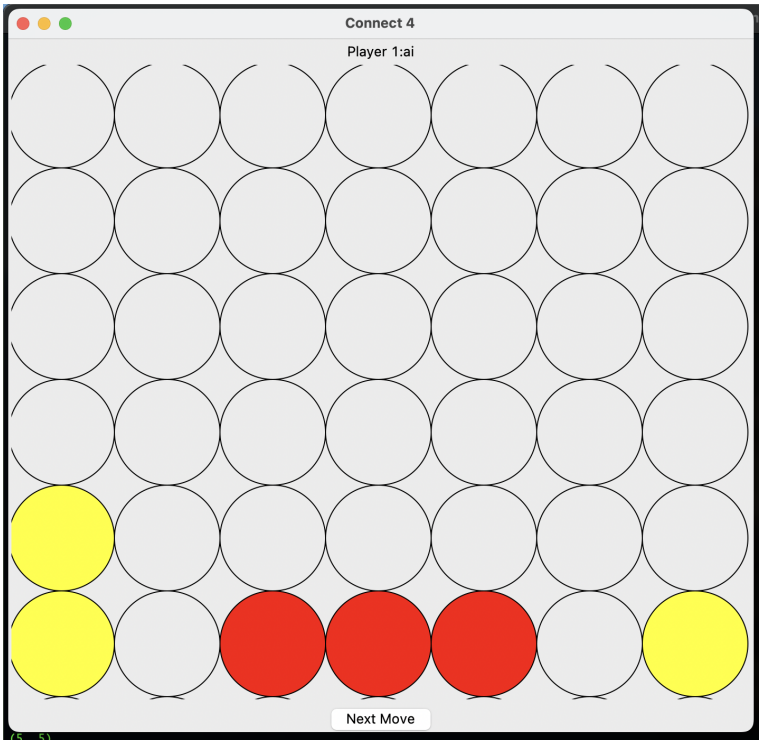

Figure 1: Snapshot from a human-ai game

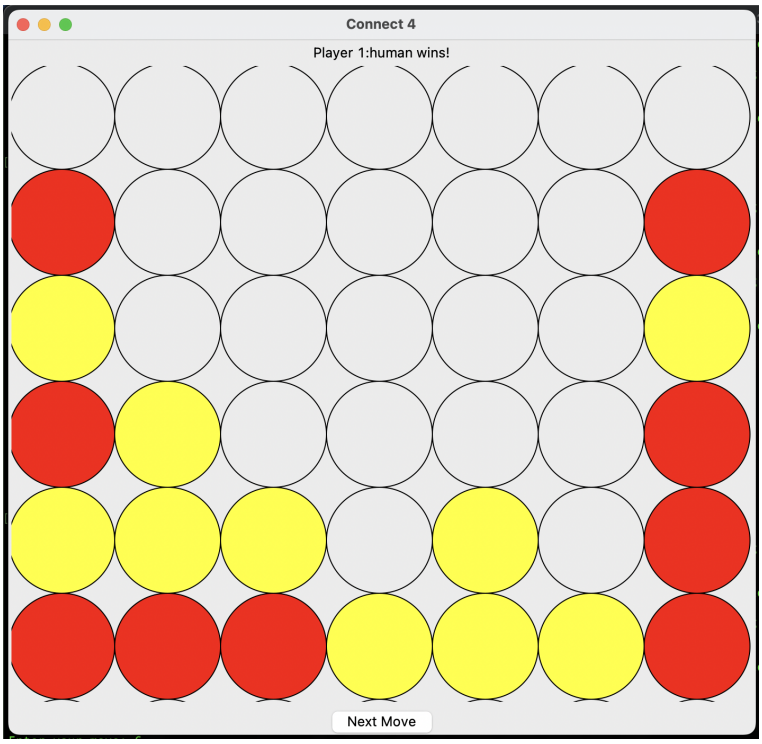

Figure 2: Snapshot from a human-ai game where I win

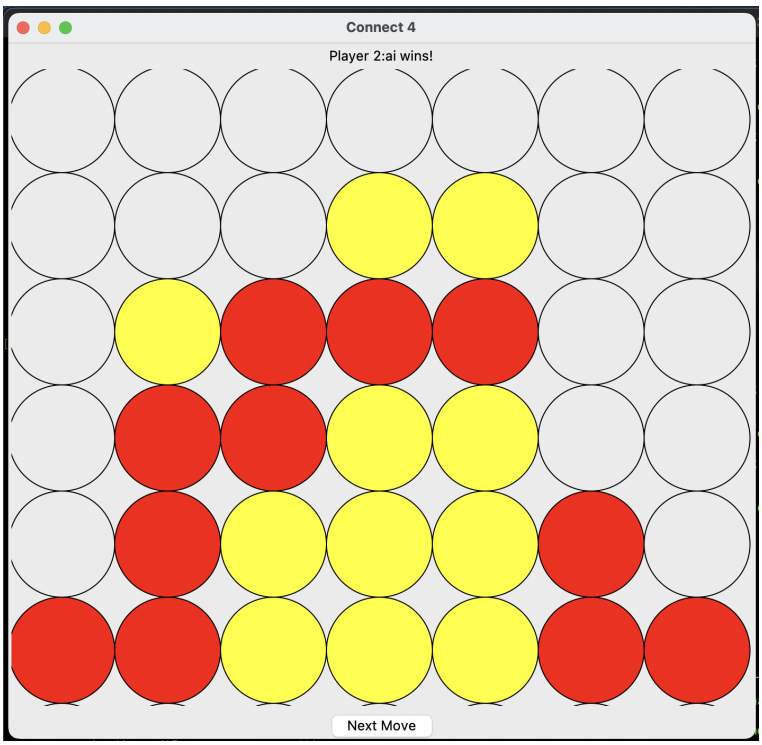

Figure 3: Snapshot from a human-ai game where my algorithm wins

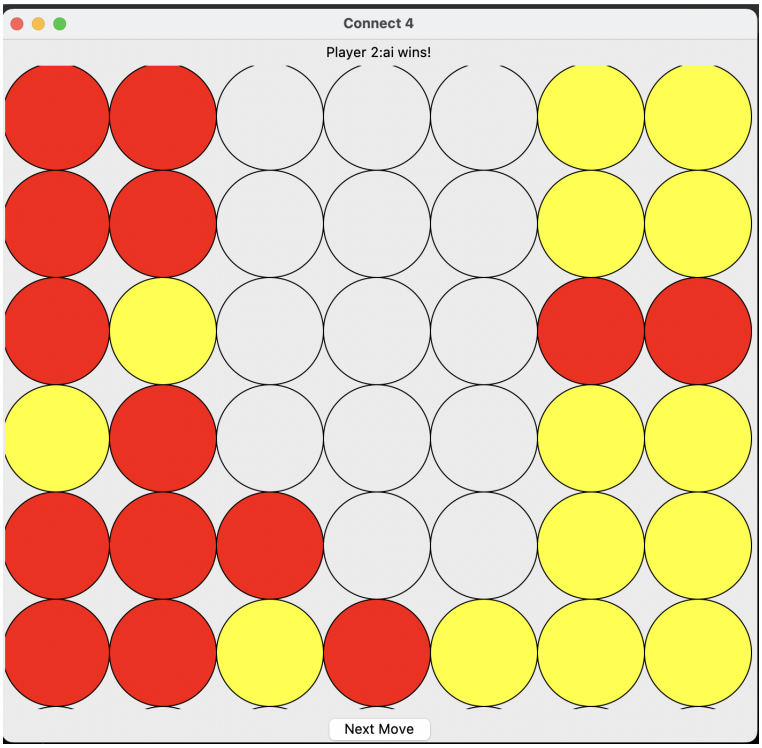

Figure 4: Snapshot from a ai-ai game

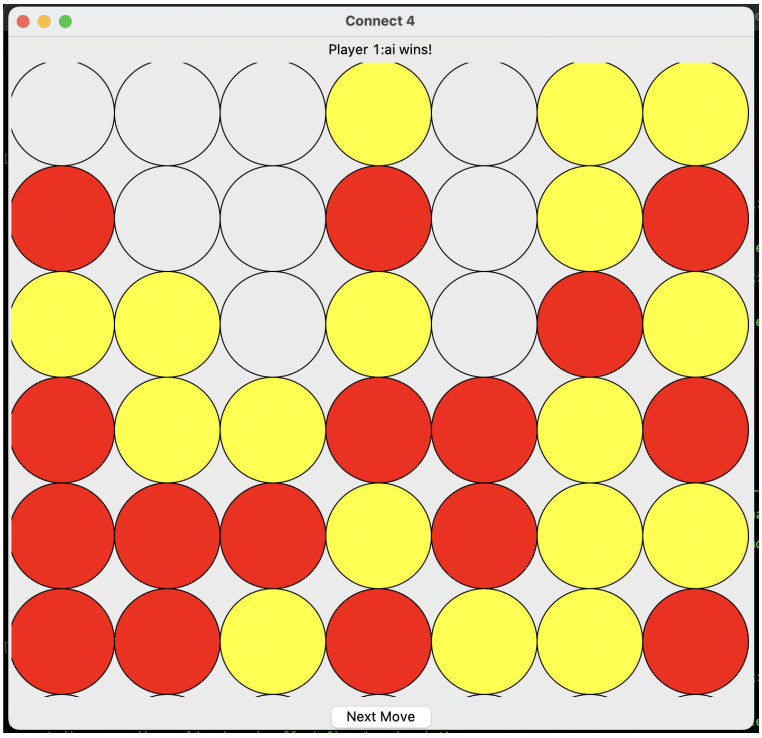

Figure 5: Snapshot from a ai-ai game