590PR Final Report

Explanations need to be added:
In class we did not have detailed explanations about MCTS, and after we consulted the related references, we found that **there are Monte Carlo Simulations during MCTS**: The precondition of calculating the UCB (Upper Confidence Bounds) values of nodes is to make the simulations of all possible game methods. Which means in every loop, the AI needs to put the stones on the whole board and simulate a complete match in the background. In the max time or max steps, it will try to simulate all the complete matches and return the number of matches and win times, then use UCB value to judge which node has better feedback.

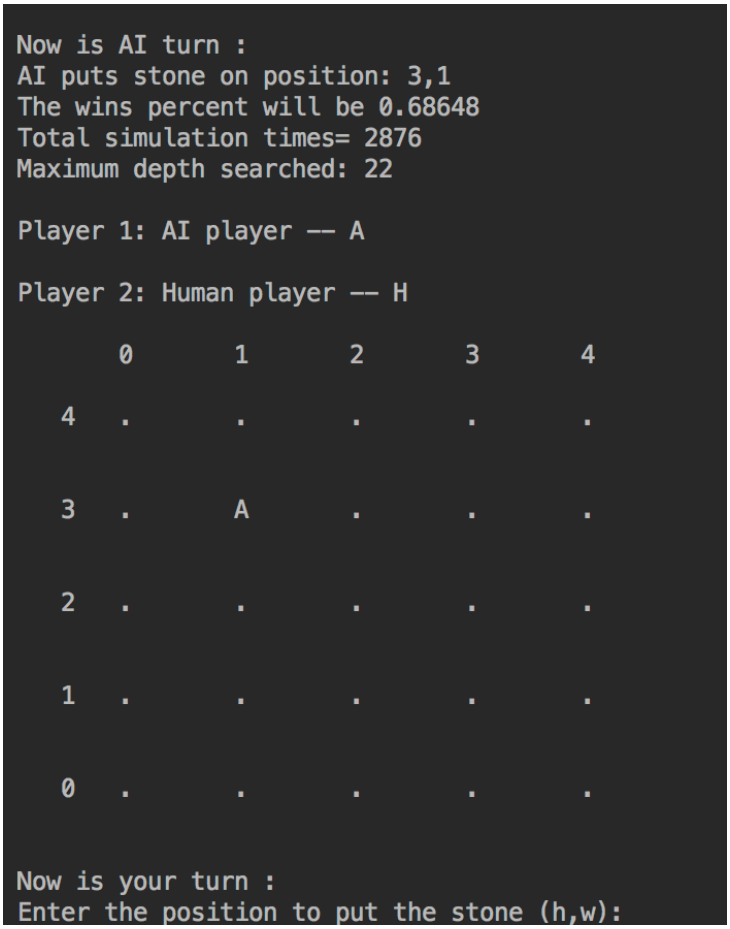
Introduction
In this project, we implemented a Four-In-Row game. At the beginning of the game, player has three choices, MCTS vs Random Simulation, MCTS vs MCTS, or play own game. Let's first choose to play our own game.

```
Welcome to Four-In-Row!
a. Do you want to compare MCTS and Random?
b. Do you want to compare two MCTS AI?
c. Start Your Own Game
Please choose (a/b/c)
c
```

After that, player can determine the size of the board and it must be larger than 4 * 4. Player can continue to choose whether to be the first player and then start the game. If player chooses to play first, it will ask you to enter a position to put the stone:

```
Now start your own game!
Please input the width/height of the board, the length need to be larger than 4
5
Do you want to play first?(y/n)
y
Player 1: AI player -- A

Player 2: Human player -- H

      0      1      2      3      4

   4  .      .      .      .      .

   3  .      .      .      .      .

   2  .      .      .      .      .

   1  .      .      .      .      .

   0  .      .      .      .      .


Now is your turn :
Enter the position to put the stone (h,w): 2,2
```

If player chooses not to play first, we run Monte Carlo simulations for AI player to play the game. We use the MCTS to implement the simulation process. The program will show the move of AI and detail information of the MCTS simulation. As pictures shows below, in the first step, the AI simulates 2876 times and chooses to put stone at (3,1), the winning percentage of this stage is around 68.6%:



After several rounds of play, our program will judge whether to end the game based on the stones on the board and will display 3 kinds of results at last including win, lose and tie.

We can use simple random simulation by changing the model_choice parameter in the Game class. If we choose to compare MCTS and Random Simulation at the beginning of the game, the program will run the game between MCTS AI and Random AI:

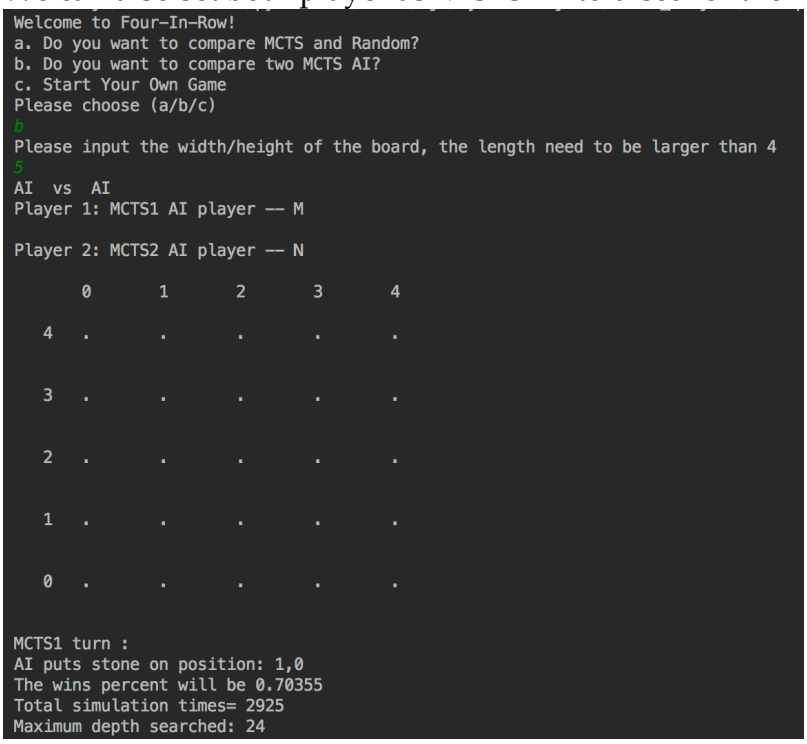
```
Welcome to Four-In-Row!
a. Do you want to compare MCTS and Random?
b. Do you want to compare two MCTS AI?
c. Start Your Own Game
Please choose (a/b/c)
a
Please input the width/height of the board, the length need to be larger than 4
5
AI  vs  Random
Player 1: MCTS AI player -- M

Player 2: Random AI player -- R

      0      1      2      3      4

   4  .      .      .      .      .

   3  .      .      .      .      .

   2  .      .      .      .      .

   1  .      .      .      .      .

   0  .      .      .      .      .


MCTS turn :
AI puts stone on position: 2,2
The wins percent will be 0.525
Total simulation times= 2920
Maximum depth searched: 15
```

Result:

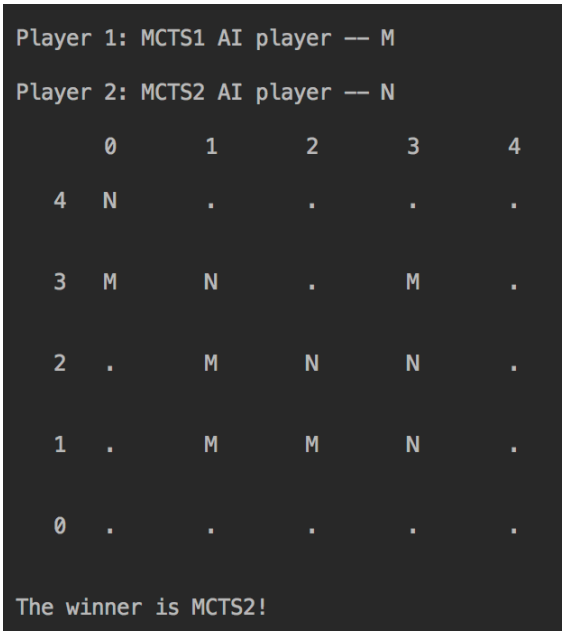
```
Player 1: MCTS AI player -- M

Player 2: Random AI player -- R

      0      1      2      3      4

   4  .      .      R      M      .

   3  .      .      R      M      .

   2  .      .      .      M      .

   1  .      .      .      M      .

   0  .      R      .      .      .


The winner is: MCTS AI!
```

The game result shows that MCTS AI always beat the random one, which means our simulation performs well and it is more convincing than random simulation.

We can also set both player as MCTS AI to discover the result:



Result:



After a number of experiments, we found that MCTS1 and MCTS2 all have chance to win the game.

Findings on the results:

1. When we run our program in 5*5 board, the simulation is good enough to calculate a position for AI to put the stone. However, with the increase of the size of the board, we can find that the performance of our program decreased. Extending the simulation time could somehow improve the performance, but the experience of playing the game would be terrible since you have to wait a long time for AI to play.
2. Since when we play MCTS1 against MCTS2, both AI can win the game, which means the order of choosing to play as the first or second player doesn't have an obvious effect on the simulation result.
3. AI doesn't keep winning the game, human players have chances to beat the AI. Possible reason is that the algorithm is not intelligent enough and we didn't have an enough simulation time.