INFO6205 Program Structures and Algorithms

Yu Lai NUID: 002640845
Ting Guo NUID: 002834835
Zenan Fan NUID: 002854067
GitHub: https://github.com/laiyumi/INFO6205_final_project.git

Spring 2024

1 Our game: Connect 4

Connect 4 is a two-player board game in which the players first choose a color and then take turns dropping colored discs into a vertically suspended 7-column, 6-row grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs with four discs.

2 MCTS

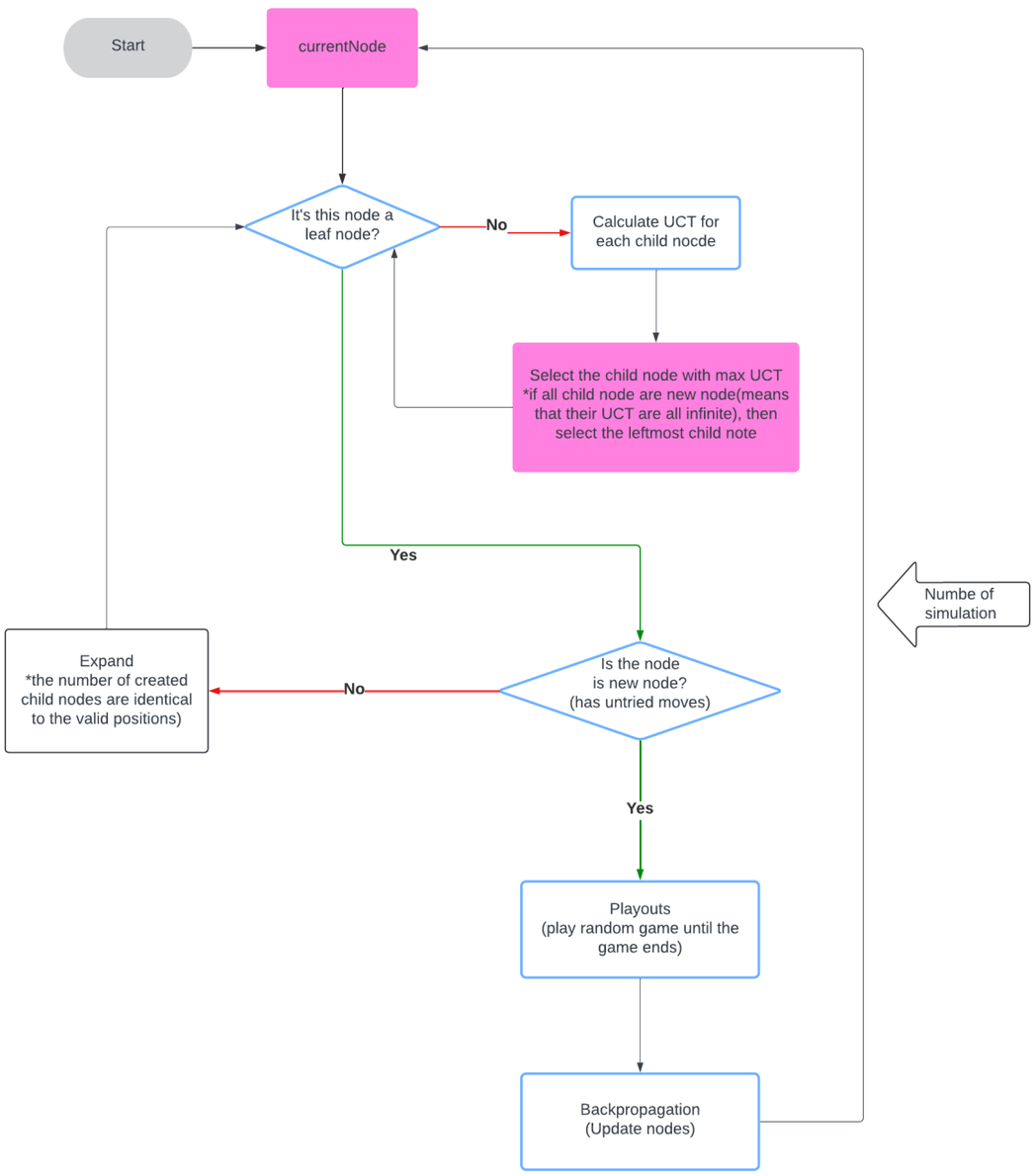

Figure 1: MCTS Workflow

3 Unit Tests

Here are the unit tests:

- Test whether the MCTS can properly find the next move on an empty board.
- Test if the game correctly identifies a winning outcome when four consecutive markers in a row by the same player.
- Test if the game correctly identifies a winning outcome when four consecutive markers in a column by the same player.
- Test if the game correctly identifies a winning outcome when four consecutive markers in a diagonal by the same player.
- Measures the amount of time taken by the MCTS to compute the next move, ensuring it is less than 1000 ms.
- Test whether the MCTS, playing as Player 1, can accurately find a winning move given a preset board state.
- Test whether the MCTS, playing as Player 2, can accurately find a winning move given a preset board state.
- Test whether the MCTS correctly identifies the winning rate for the next move of Player 1.

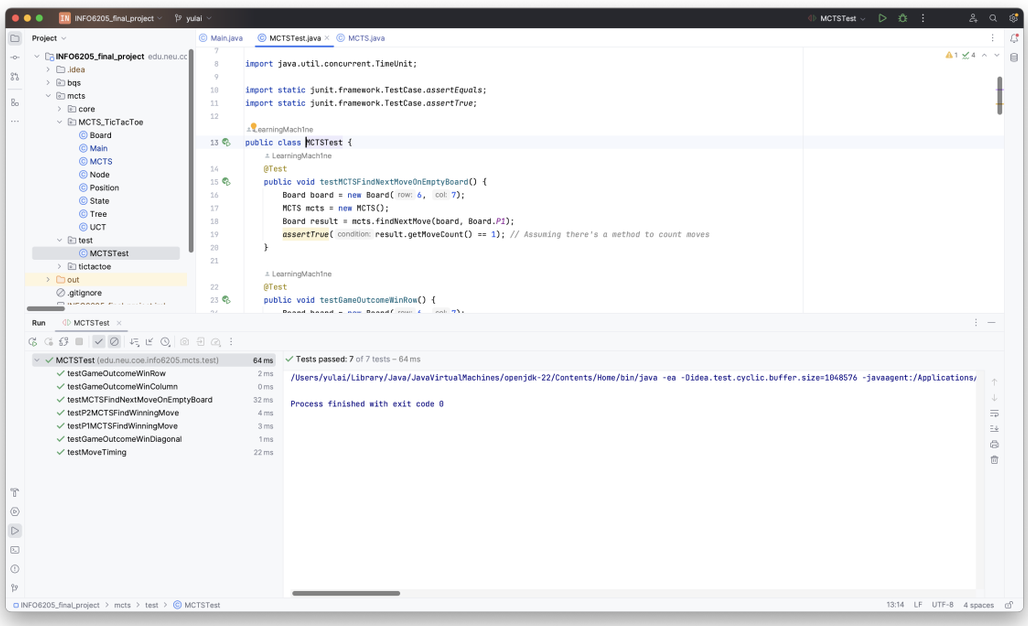

Figure 2: Unit Tests Evidence

4 Timing of Runs

To assess the performance of our program, we conduct benchmarking at different scales by choosing different number of simulation in MCTS and running the program 50, 100, 200, and 400 times. The results and average timings are presented below:

Robot VS. Robot Mode (MCTS simulation = 100)

Runs	N = 50	N = 100	N = 200	N = 400
Player 1 (opener)	25	55	110	206
Player 2	25	45	90	193
Draw	0	0	0	1
Average Time per game	9.37ms	8.53ms	8.195ms	8.4025 ms
Player 1: Average Time per game	9.9ms	9.24 ms	8.8ms	8.98 ms
Player 2: Average Time per game	8.84ms	7.82 ms	7.59ms	7.825 ms

Robot VS. Robot Mode (MCTS simulation = 500)

Runs	N = 50	N = 100	N = 200	N = 400
Player 1 (opener)	26	57	106	223
Player 2	24	43	94	177
Draw	0	0	0	0
Average Time per game	61.81 ms	61.605ms	62.912ms	63.91ms
Player 1: Average Time per game	65.04 ms	64.75 ms	65.785 ms	66.9625 ms
Player 2: Average Time per game	58.58 ms	58.46 ms	60.04 ms	60.8725 ms

Robot VS. Robot Mode (MCTS simulation = 1000)

Runs	N = 50	N = 100	N = 200	N = 400
Player 1 (opener)	35	55	118	209
Player 2	15	43	82	188
Draw	0	2	0	3
Average Time per game	129.57ms	135.925 ms	137.0225 ms	138.089ms
Player 1: Average Time per game	136.1 ms	142.38 ms	143.465 ms	144.385 ms
Player 2: Average Time per game	123.06 ms	129.47 ms	130.58 ms	131.795 ms

Robot VS. Robot Mode (MCTS simulation = 2000)

Runs	N = 50	N = 100	N = 200	N = 400
Player 1 (opener)	25	58	97	213
Player 2	25	41	101	183
Draw	0	1	2	4
Average Time per game	275.09	282.15 ms	284.71 ms	285.236ms
Player 1: Average Time per game	287.96 ms	294.74 ms	296.335 ms	297.1675 ms
Player 2: Average Time per game	262.22 ms	269.56 ms	273.085 ms	273.305 ms

Figure 3: Timing and Result of Runs

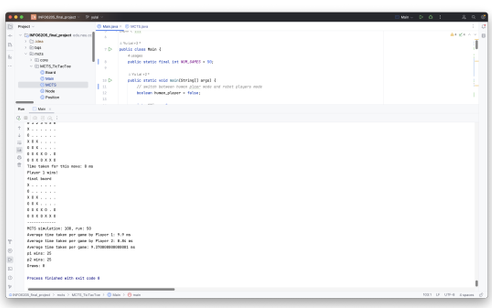

Figure 4: 50 runs with MCTS 100 simulations

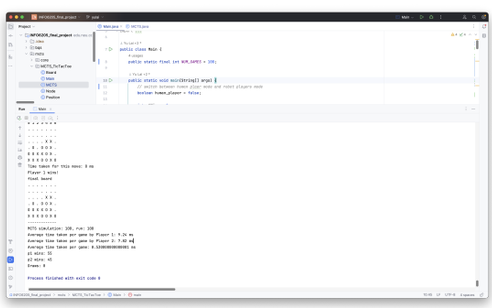

Figure 5: 100 runs with MCTS 100 simulations

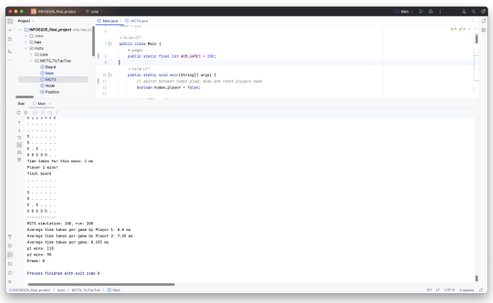

Figure 6: 50 runs with MCTS 100 simulations

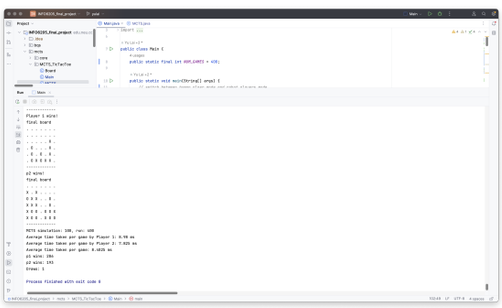

Figure 7: 100 runs with MCTS 100 simulations

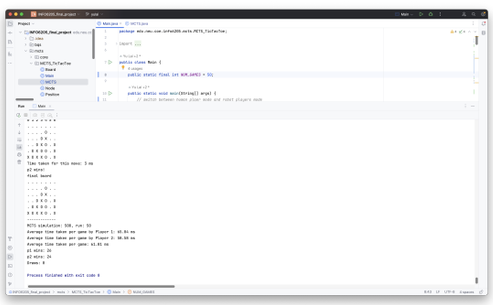

Figure 8: 50 runs with MCTS 100 simulations

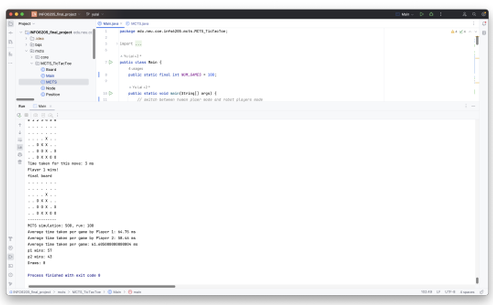

Figure 9: 100 runs with MCTS 100 simulations

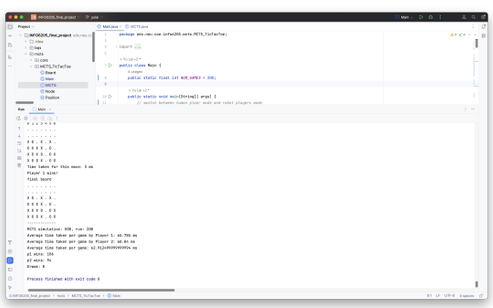

Figure 10: 50 runs with MCTS 100 simulations

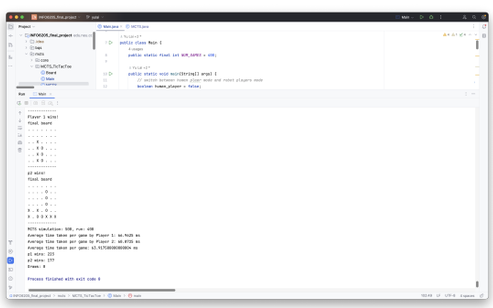

Figure 11: 100 runs with MCTS 100 simulations

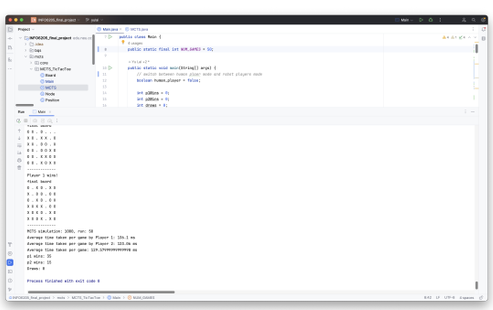

Figure 12: 50 runs with MCTS 100 simulations

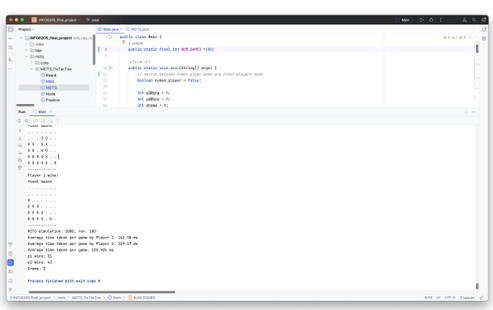

Figure 13: 100 runs with MCTS 100 simulations

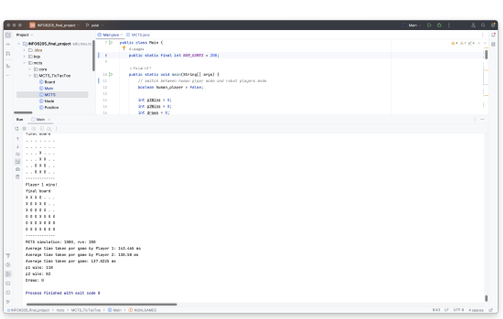

Figure 14: 50 runs with MCTS 100 simulations

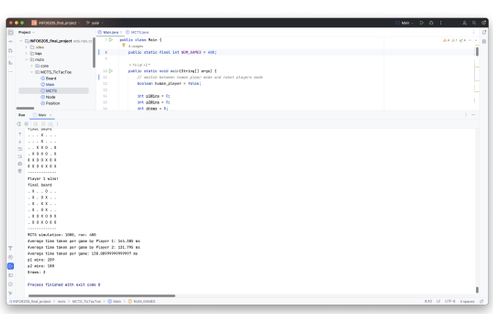

Figure 15: 100 runs with MCTS 100 simulations

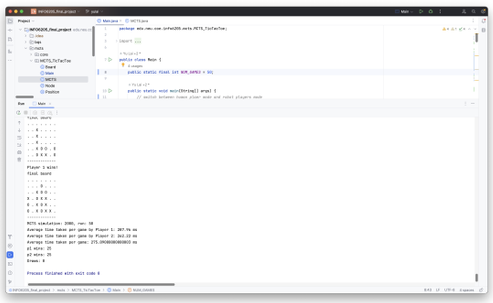

Figure 16: 50 runs with MCTS 100 simulations

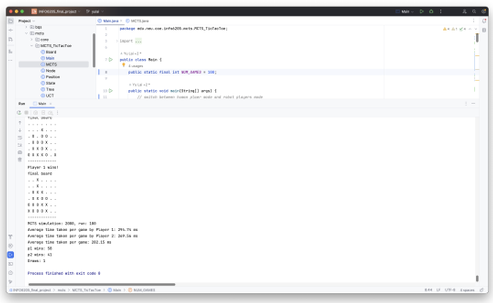

Figure 17: 100 runs with MCTS 100 simulations

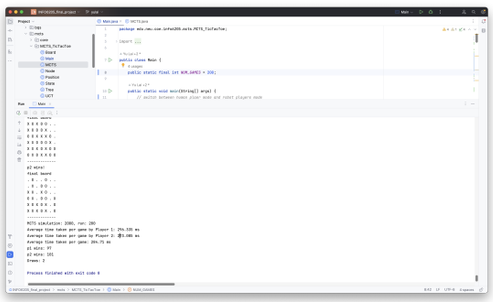

Figure 18: 50 runs with MCTS 100 simulations

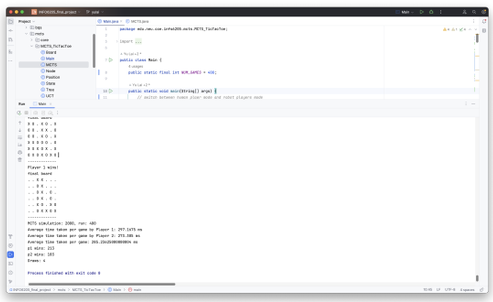

Figure 19: 100 runs with MCTS 100 simulations

5 Guidelines to Run

1. **Clone the repository to your local machine**
   Use the following command:

   ```
   git clone https://github.com/laiyumi/INFO6205_final_project.git
   ```

2. **Navigate to the repository folder**
   Navigate to the cloned repository folder in your terminal or command line using:

   ```
   cd INFO6205_final_project
   ```

3. **Open the repository with IntelliJ IDEA**
   Navigate to Main class and run.

4. **Test Connect 4 code**

   ```
   cd test
   ```

   Navigate to MCTSTest class and run.

6 Live Demo

Demo(Click me)

7 Observations and Analysis

This table displays the winning rates for two players across various MCTS simulation parameters and numbers of games played. Generally, the winning rates fluctuate between 40%-60%, a reflection of both players employing the MCTS algorithm to optimize their moves at each turn. As the number of MCTS simulations rises, the winning rates for both players tend to converge towards 50%. This indicates that with more computational depth in decision-making, the skill levels of the players are increasingly matched, leading to more evenly balanced games. Additionally, there appears to be a trend where the frequency of drawn games increases in conjunction with the number of simulations and the number of games played.

Average Winning Rate

	N = 50	N = 100	N = 200	N = 400
	MCTS simulation = 100			
Player 1	50%	55%	55%	51.5%
Player 2	50%	45%	45%	48.25%
Draw	0%	0%	0%	As
	MCTS simulation = 500			
Player 1	52%	57%	53%	55.75%
Player 2	48%	43%	47%	44.25%
Draw	0%	0%	0%	0%
	MCTS simulation = 1000			
Player 1	70%	55%	59%	52.25%
Player 2	30%	43%	41%	47%
Draw	0%	2%	0%	0.75%
	MCTS simulation = 2000			
Player 1	50%	58%	48.5%	53.25%
Player 2	50%	41%	50.5%	45.75%
Draw	0%	1%	1%	1%

Figure 20: Unit Tests Evidence

Here is the performance of each player:

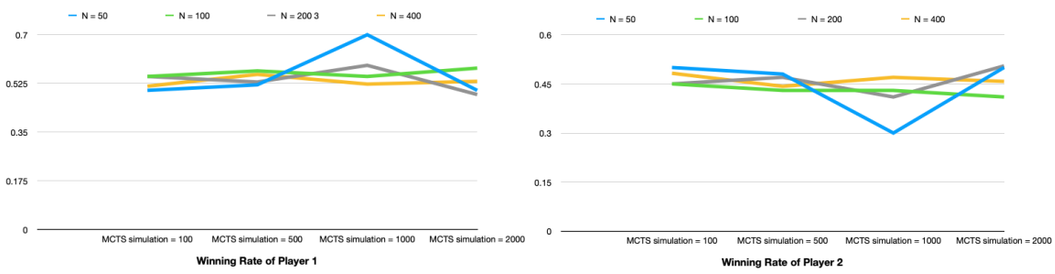

Figure 21: Unit Tests Evidence

Player 1: The peak winning rate for Player 1 is with MCTS simulation = 1000, particularly for N = 50 games. The general trend suggests that increasing the number of simulations improves Player 1's performance significantly up to this point, after which the benefits diminish.

Player 2:The highest winning rate is at the highest level of MCTS simulation when N = 200. In most cases, player 2 has a winning rate below 50% and even drops to 30% with the simulation 1000 and N=50.

Analysis by Number of Games (N): For N = 50, both players show improved performance with more simulations, but Player 1 more so than Player 2, indicating Player 1 may utilize early game states more efficiently. At N = 100 and N = 200, Player 1's performance peaks at MCTS = 1000 simulations, suggesting that at this level, the algorithm's exploration-exploitation balance may be optimized for the complexity of Connect 4. At N = 400, Player 1's performance dips at the highest simulation count, possibly due to overfitting or diminishing returns on the computational effort.

8 Conclusions

Since both players use the same MCTS algorithm to do the move, the variation in winning rates suggests that other factors influence the outcomes.

Randomness in MCTS: The MCTS algorithm includes a degree of randomness in its simulation phase, which can lead to different game outcomes even with identical algorithms.

First-Move Advantage: If one player consistently has the first move, this could skew the winning rates, as Connect 4 is known to have a built-in advantage for the first player with perfect play.

Statistical Variation: The differences could be due to statistical variation, especially in scenarios with fewer games (N = 50 or N = 100), where the sample size may not be large enough to average out anomalies.

Algorithm Parameters: Even if the MCTS algorithm itself is the same, the parameters such as the exploration factor in the UCT (Upper Confidence bounds applied to Trees) formula could influence how exploration versus exploitation is balanced.

In conclusion, while both players use the same MCTS algorithm, various factors such as randomness, statistical variance, and potential first-move advantages may contribute to the observed outcomes. Additionally, the effectiveness of MCTS is not strictly proportional to the number of simulations, especially considering the potential for overfitting or computational constraints.

9 Citatioins

Monte Carlo Tree Search (Wikipedia article)
Connect Four (Wikipedia article)
Explaining Monte Carlo Tree Search - AlphaGo's Core Algorithm(Youtube)