Program Structures and Algorithms

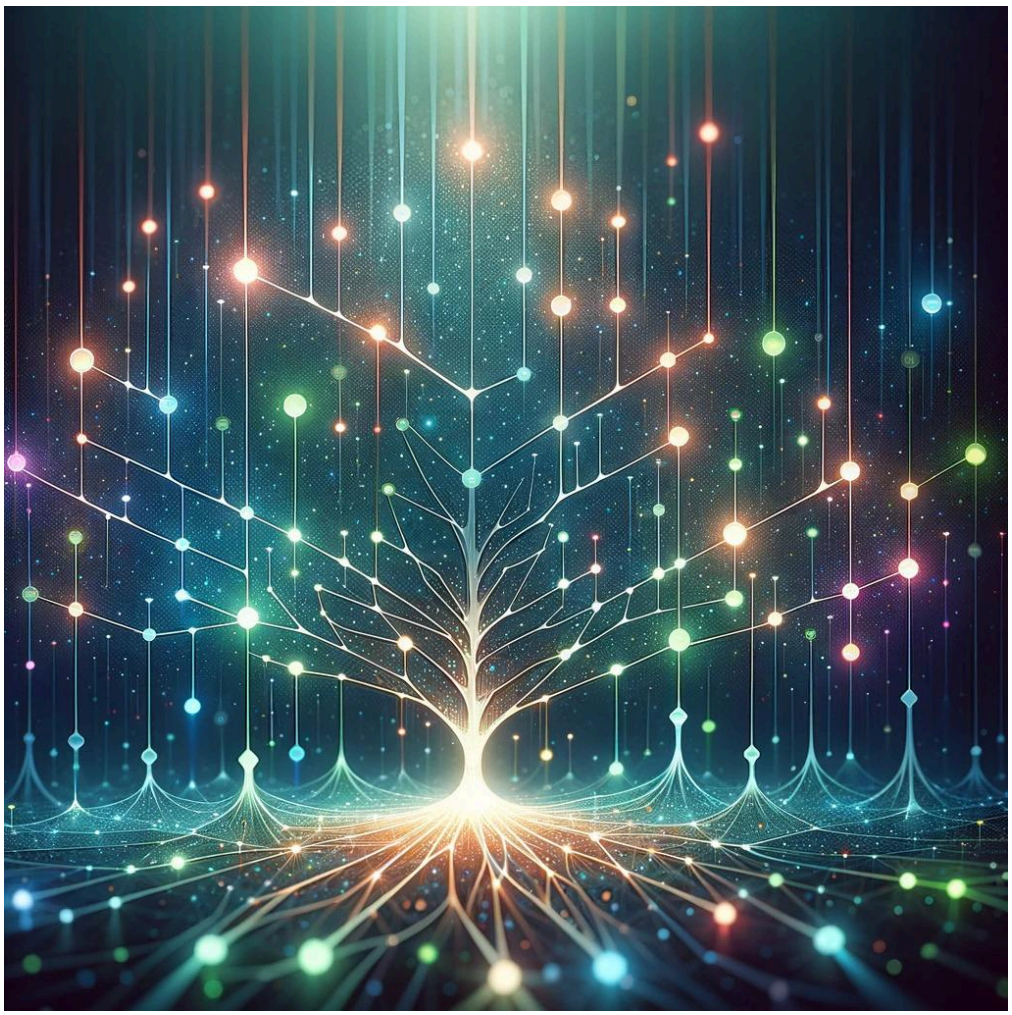
Spring 2024

FINAL PROJECT REPORT

AUTHORS: Kartikey Vijayakumar Hebbar (002276938), Meet Karnik (002795334)

GITHUB LINK: https://github.com/kartikeyhebbar/INFO6205-Final-Project

MONTE CARLO TREE SEARCH



MONTE CARLO TREE SEARCH

Concept & Approach:

The Monte Carlo Tree Search (MCTS) algorithm is a decision-making algorithm used primarily in artificial intelligence (AI) applications, particularly in games and situations that require optimal decision-making under uncertain conditions.

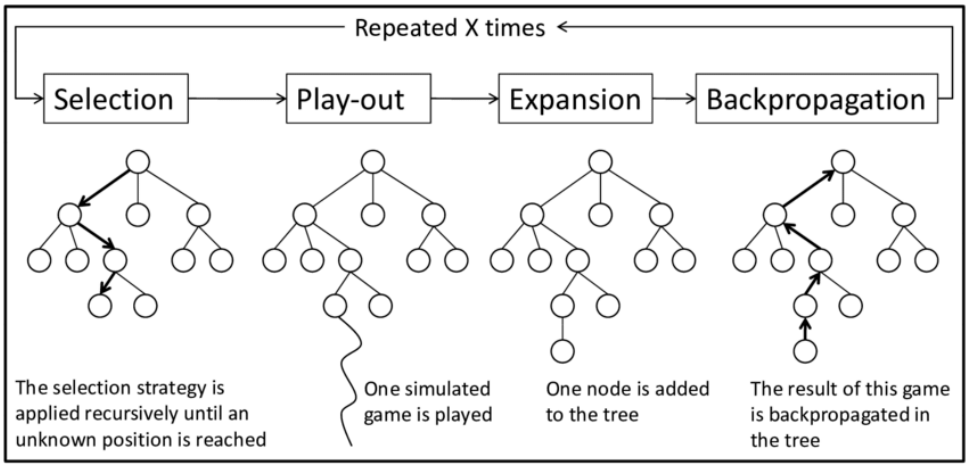
Here's a breakdown of how it works:

1. **Selection:** The algorithm starts at the root node and selects successive child nodes down to a leaf node (following Depth-First Search approach). The selection is based on a policy that balances exploration of less visited nodes with exploitation of nodes known to have a high win ratio.

2. **Expansion:** Unless the leaf node ends the game (e.g., checkmate or win/loss in a board game), it adds one or more child nodes to expand the search tree, based on available moves.

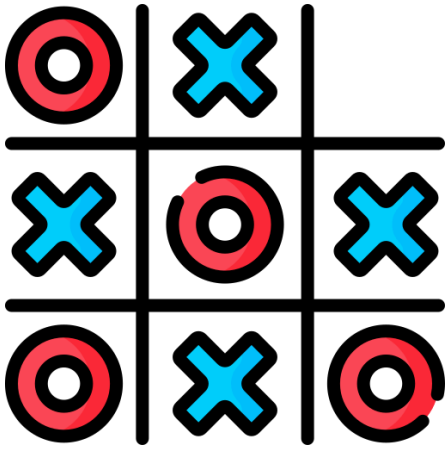
3. **Simulation:** From the new nodes, the algorithm simulates random games (also known as playouts or rollouts). The outcomes of these simulations are used to estimate the node values.

4. **Backpropagation:** The results of the simulations are then propagated back through the tree, updating the relevant nodes with the simulation results to improve the accuracy of their value estimates.



IMPLEMENTATION

1. TIC-TAC-TOE



Tic-tac-toe is a paper-and-pencil game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. It is a solved game, with a forced draw assuming best play from both players.

Our implementation of the Tic-Tac-Toe is a computer simulated two-player game which leverages the Monte-Carlo Tree Search (MCTS) algorithm.

There are few steps involved in the process of making any move by the CPU player:

Selection: The algorithm starts with a root node and selects a child node such that it picks the node with maximum win rate. The idea is to keep selecting optimal child nodes until we reach the leaf node of the tree. A good way to select such a child node is to use **UCT (Upper Confidence Bound applied to trees) formula**:

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}}$$

In which,

wi = number of wins after the i-th move

ni = number of simulations after the i-th move

c = exploration parameter (theoretically equal to √2)

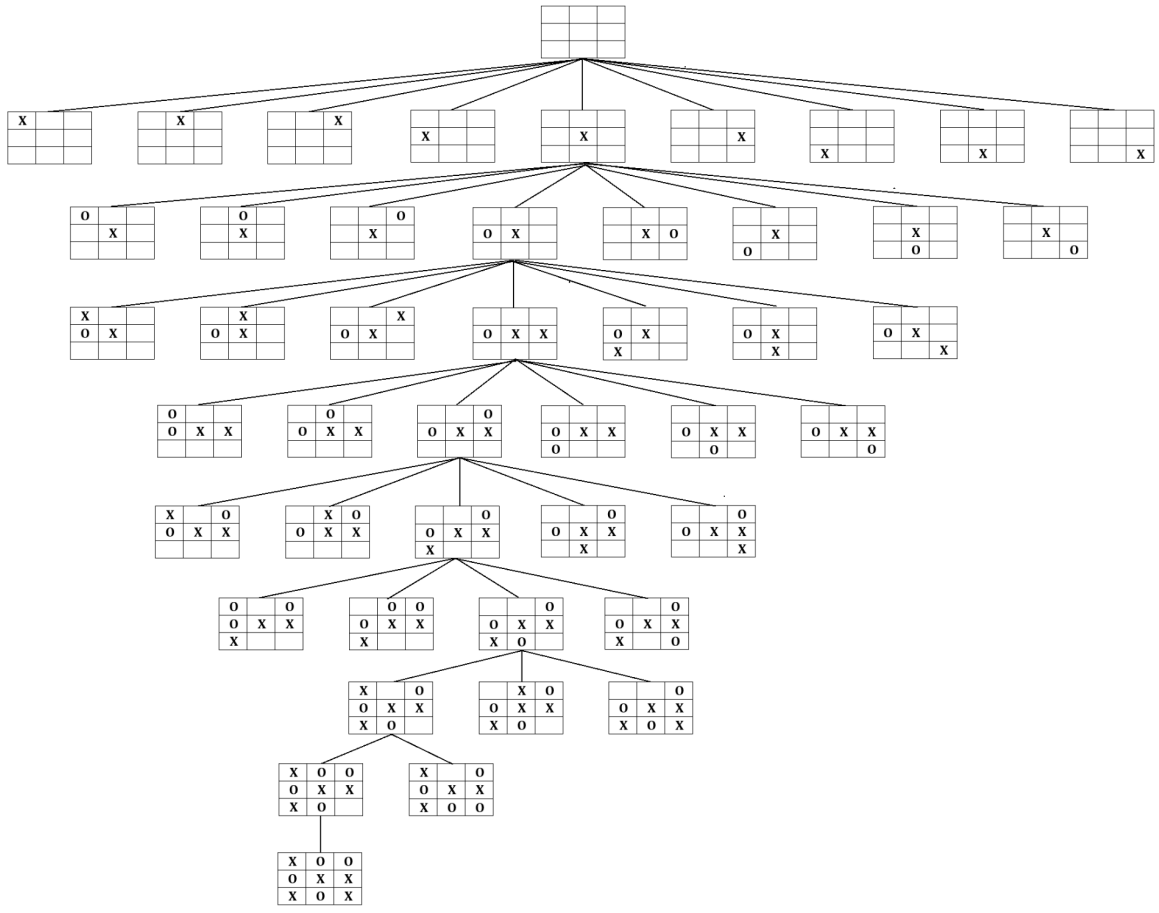
t = total number of simulations for the parent node

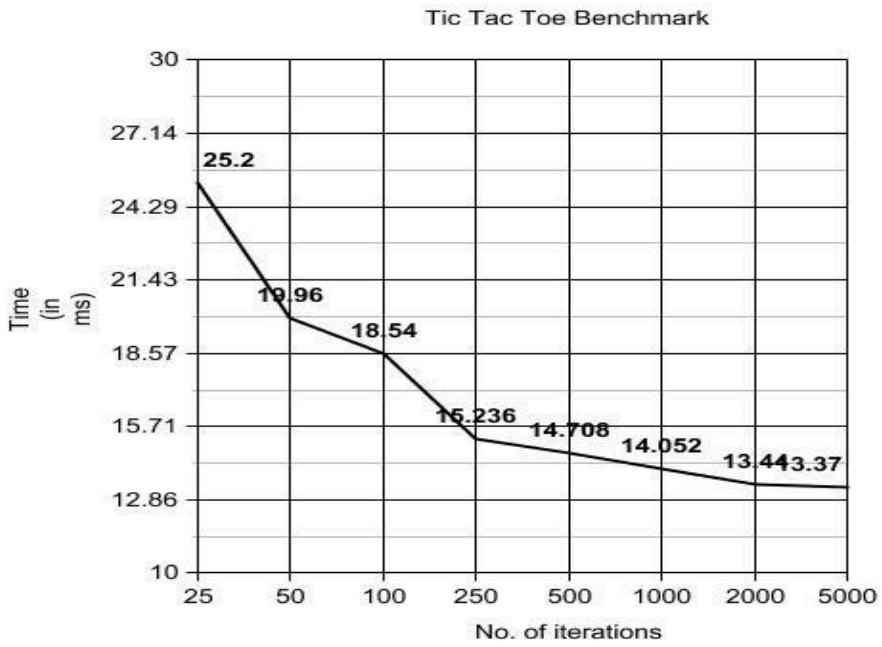
Expansion: When it can no longer apply UCT to find the successor node, it expands the game tree by appending all possible states from the leaf node.

Simulation: The algorithm picks a child node arbitrarily, and it simulates a randomized game from the selected node until it reaches the resulting state of the game. If nodes are picked randomly or semi-randomly during the play out, it is called light play out.

Backpropagation: This is also known as an update phase. Once the algorithm reaches the end of the game, it evaluates the state to figure out which player has won. It traverses upwards to the root and increments visit score for all visited nodes. It also updates the win score for each node if the player for that position has won the playout.

MCTS keeps repeating these four phases until some fixed number of iterations.



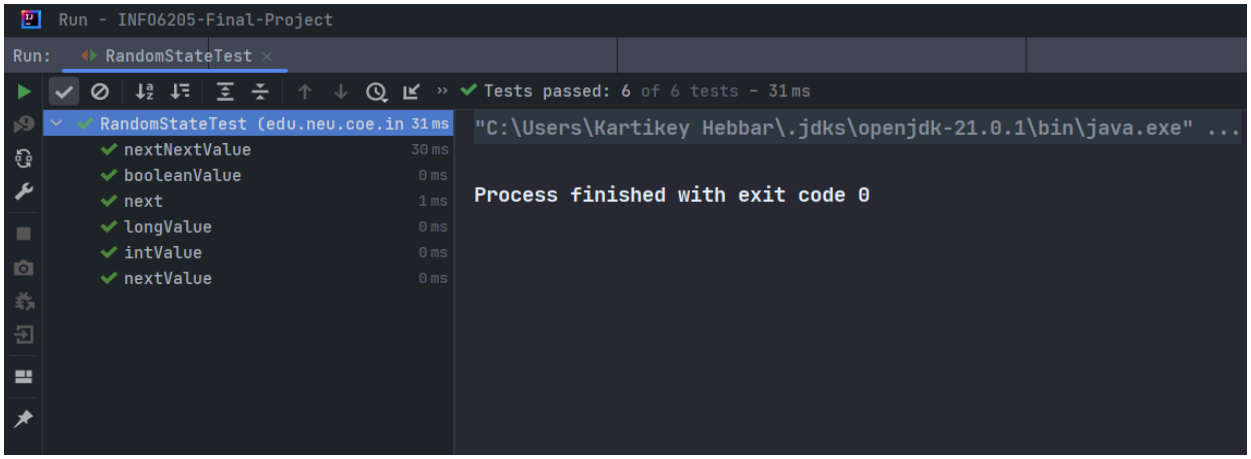


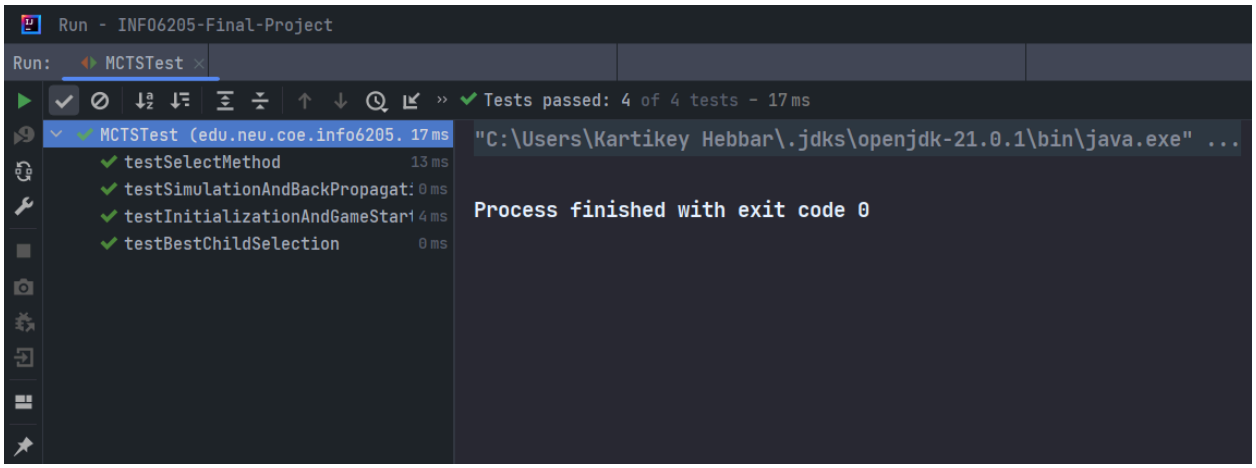
Results: The above graph shows the average time taken for MCTS to win a TicTacToe game vs the number of iterations, also based on the number of iterations, we've compiled the following table which shows the amount of times X and O have won the game. Please note that all times shown are in milliseconds(ms)

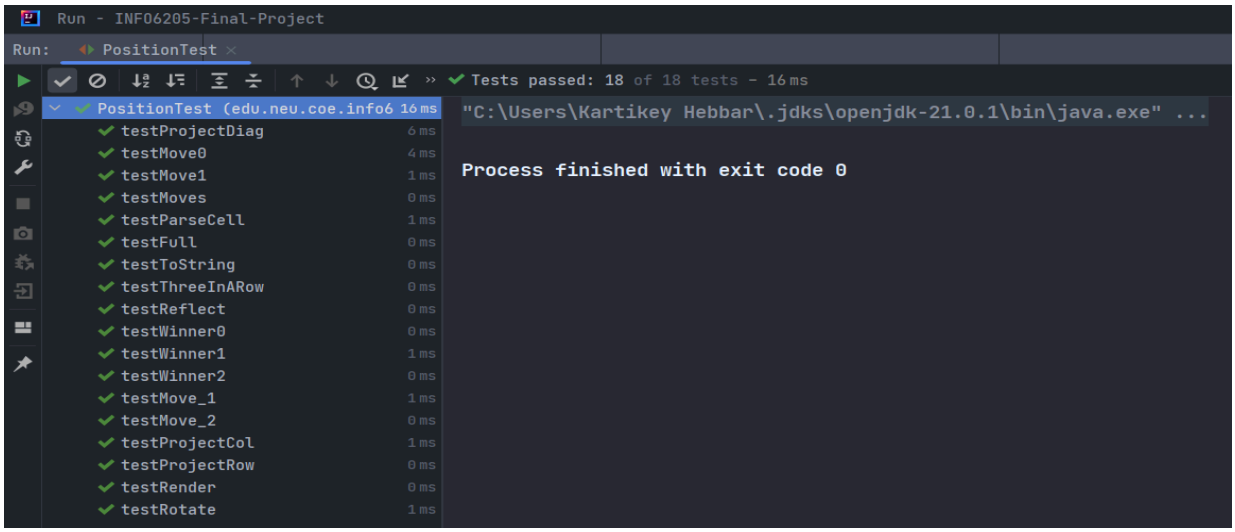
n	No. of wins(X)	No. of wins(O)	Average time taken(ms)
5000	641	4359	13.37
2000	245	1755	13.44
1000	113	887	14.052
500	57	443	14.708
250	24	226	15.236
100	10	90	18.54
50	6	44	19.96
25	1	24	25.2

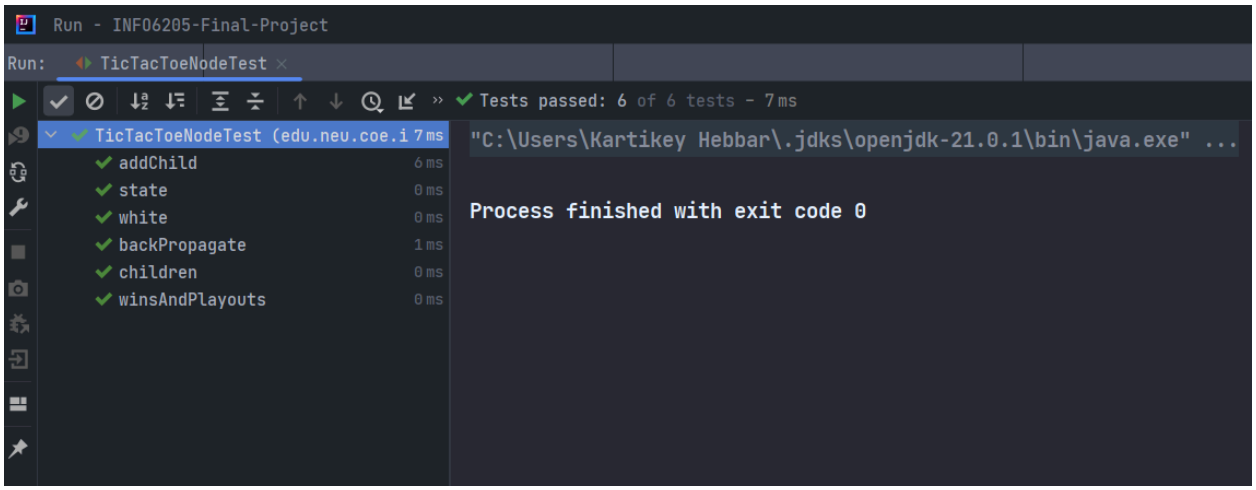
OUTPUT SNAPSHOTS

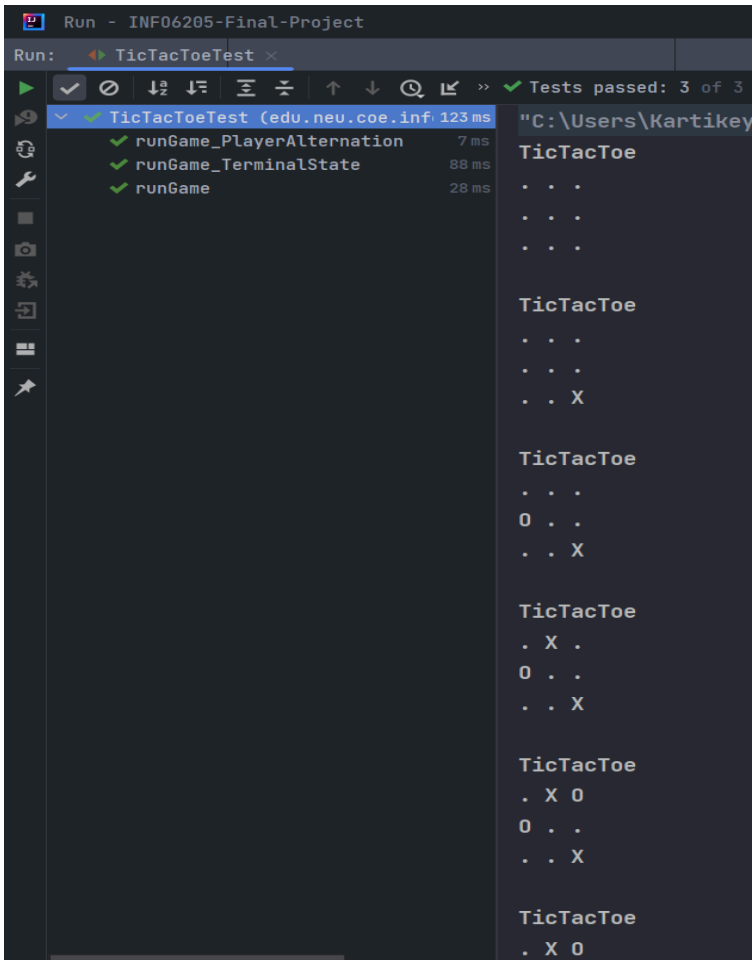
UNIT TEST SNAPSHOTS



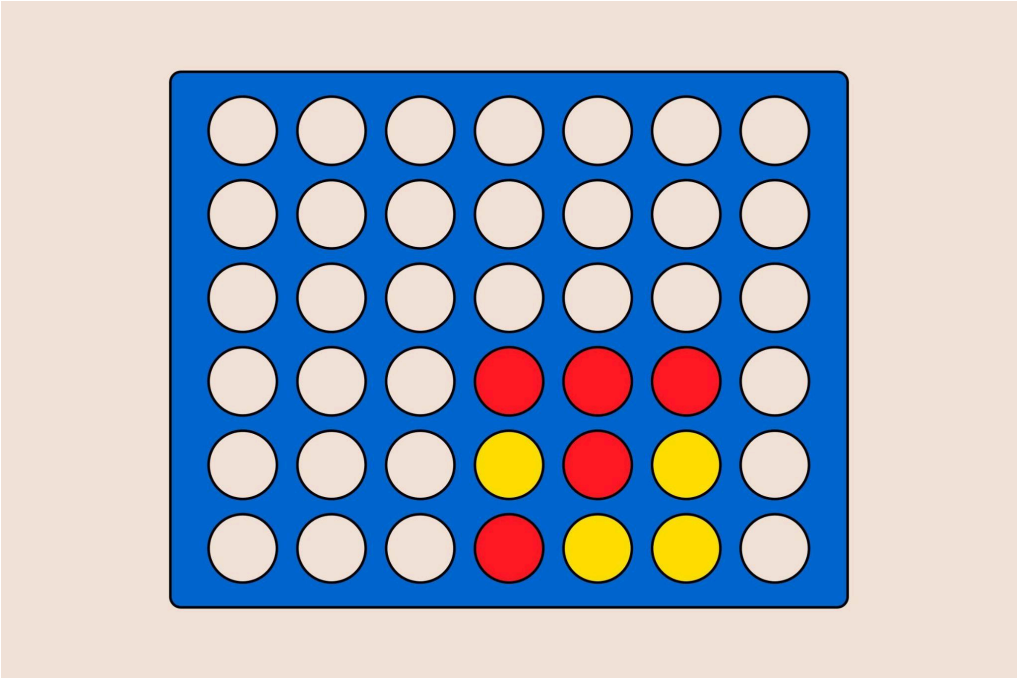





Run - INFO6205-Final-Project
Run: TicTacToeNodeTest
Tests passed: 6 of 6 tests – 7 ms
TicTacToeNodeTest (edu.neu.coe.i 7 ms
addChild 6 ms
state 0 ms
white 0 ms
backPropagate 1 ms
children 0 ms
winsAndPlayouts 0 ms
"C:\Users\Kartikey Hebbar\.jdks\openjdk-21.0.1\bin\java.exe" ...
Process finished with exit code 0


Run - INFO6205-Final-Project
Run: TicTacToeTest
Tests passed: 3 of 3
TicTacToeTest (edu.neu.coe.inf 123 ms
runGame_PlayerAlternation 7 ms
runGame_TerminalState 88 ms
runGame 28 ms
"C:\Users\Kartikey
TicTacToe
. . .
. . .
. . .
TicTacToe
. . .
. . .
. . X
TicTacToe
. . .
O . .
. . X
TicTacToe
. X .
O . .
. . X
TicTacToe
. X O
O . .
. . X
TicTacToe
. X O

2. CONNECT 4

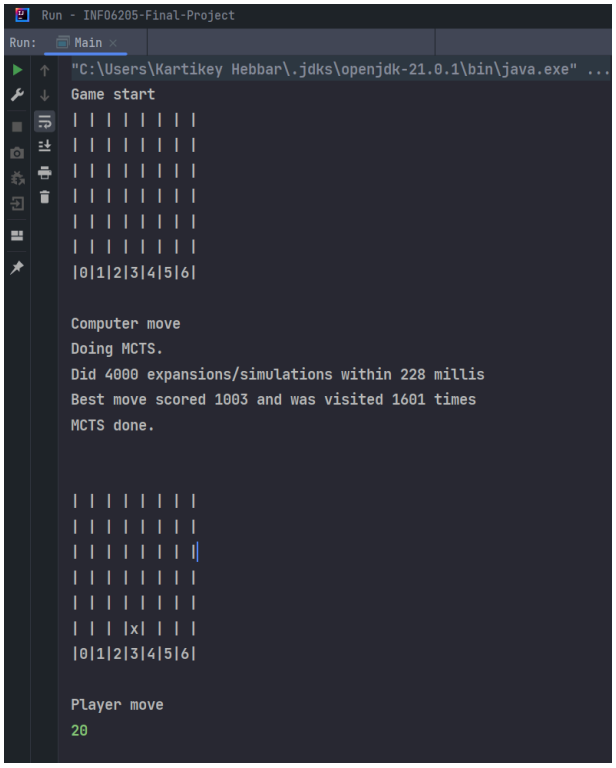


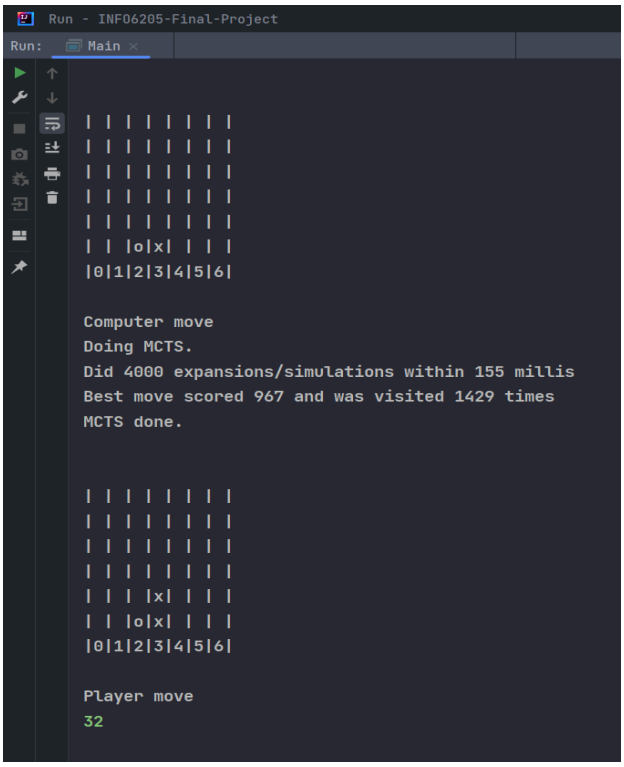
Connect Four is a game in which the players choose a color/symbol and then take turns dropping their tokens into a six-row, seven-column vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own tokens.

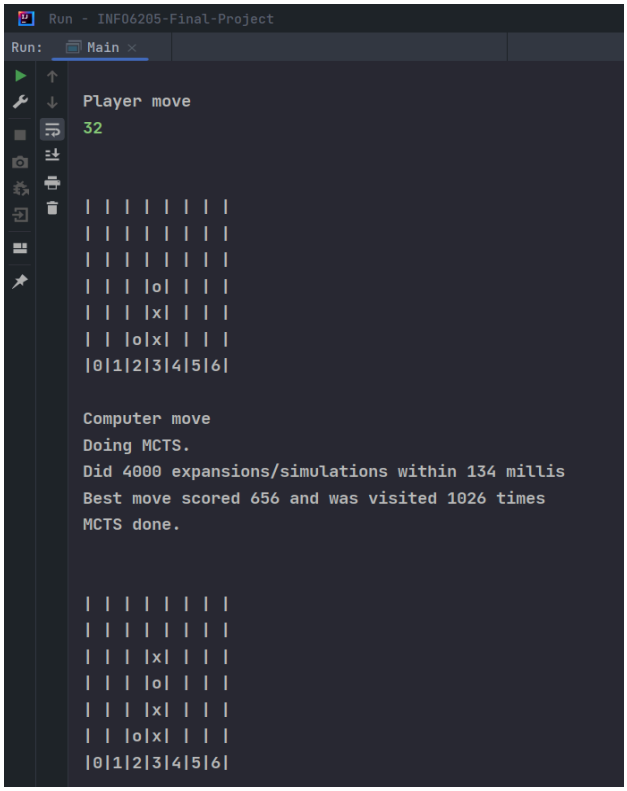
We implemented the Connect 4 game as a 2 player (Human vs CPU) gameplay. The CPU starts the game by drawing an X on the board followed by the Human drawing an O on the game board.

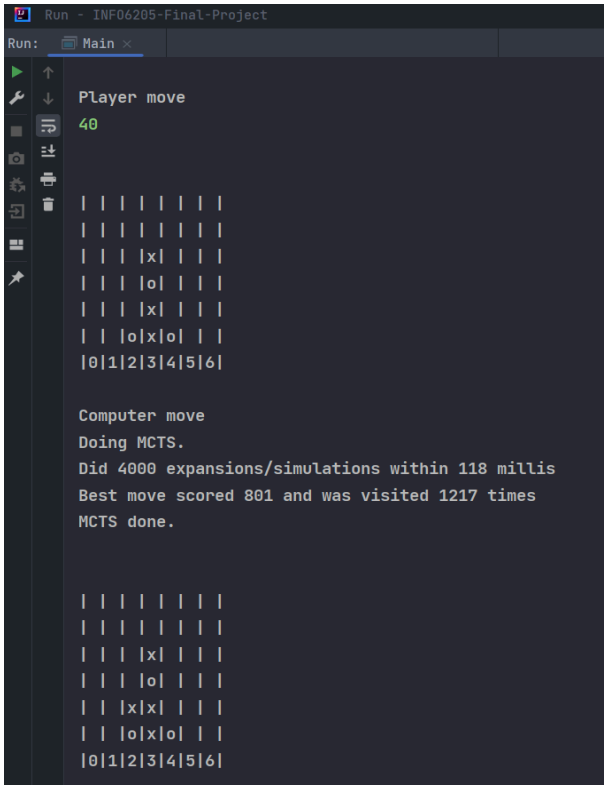
For every CPU operation, MCTS algorithm is used where it follows all the four steps as mentioned above.

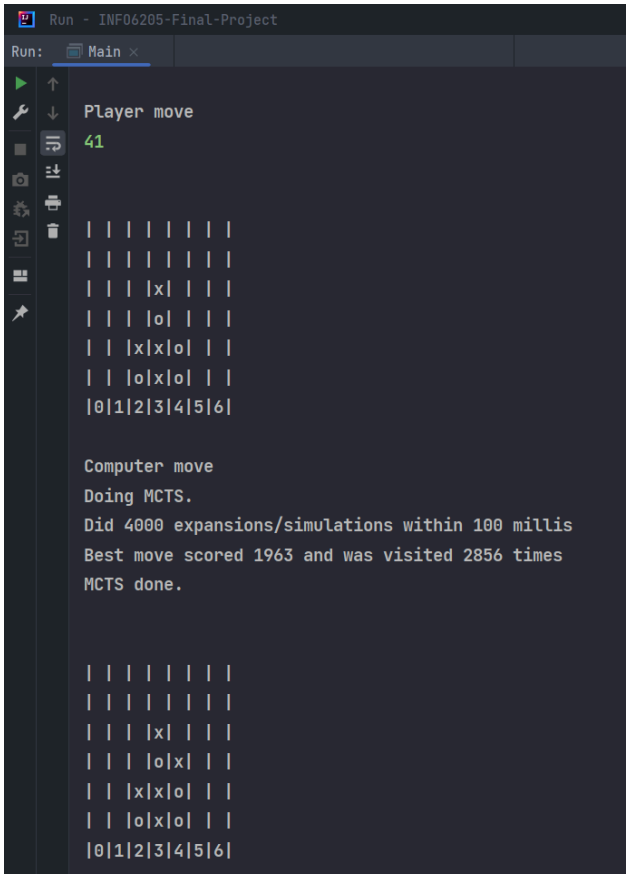
GAME SIMULATION SNAPSHOTS: MCTS IN OPERATION



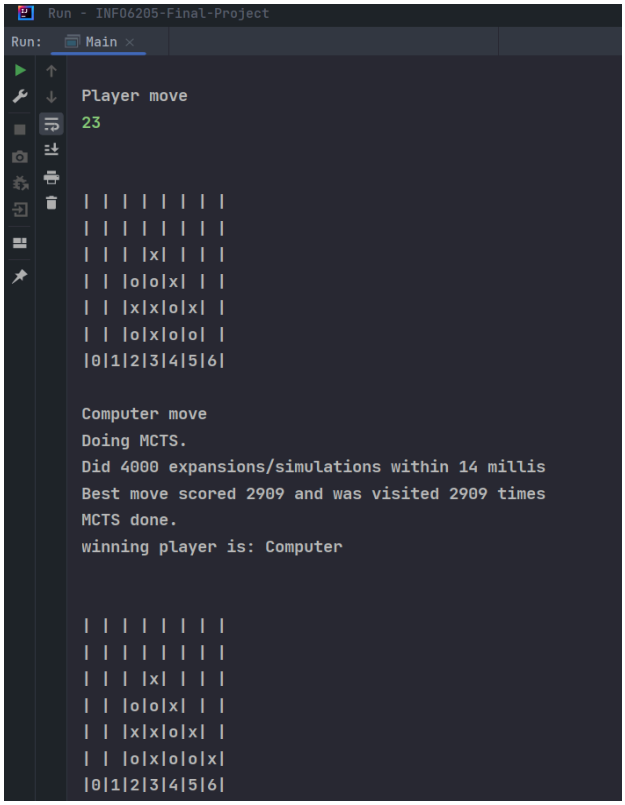


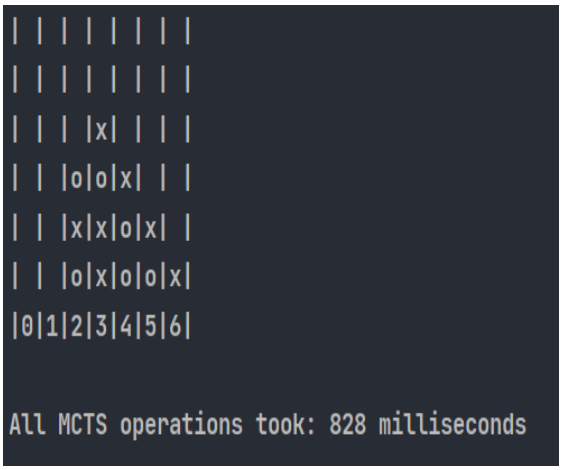


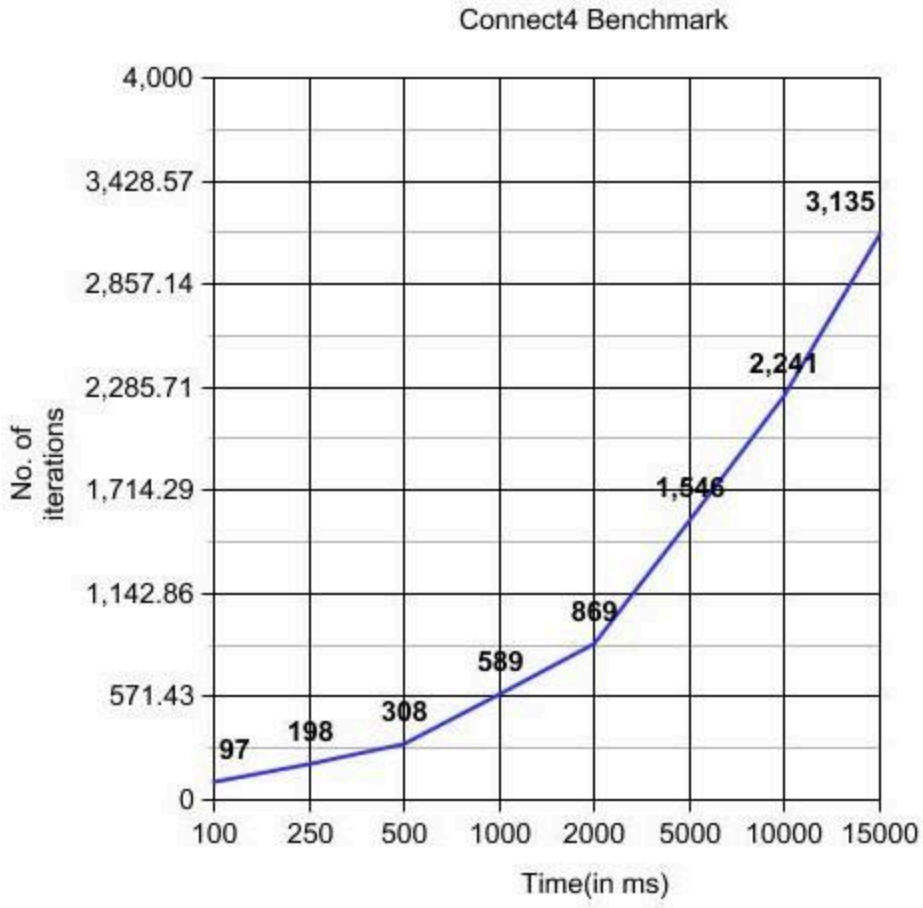



Run - INFO6205-Final-Project
Run: Main
Player move
41
			x			
			o			
		x	x	o		
		o	x	o		
0	1	2	3	4	5	6
Computer move
Doing MCTS.
Did 4000 expansions/simulations within 100 millis
Best move scored 1963 and was visited 2856 times
MCTS done.
			x			
			o	x		
		x	x	o		
		o	x	o		
0	1	2	3	4	5	6


Run - INFO6205-Final-Project
Run: Main
Player move
50
			x			
			o	x		
		x	x	o		
		o	x	o	o	
0	1	2	3	4	5	6
Computer move
Doing MCTS.
Did 4000 expansions/simulations within 79 millis
Best move scored 747 and was visited 1148 times
MCTS done.
			x			
			o	x		
		x	x	o	x	
		o	x	o	o	
0	1	2	3	4	5	6

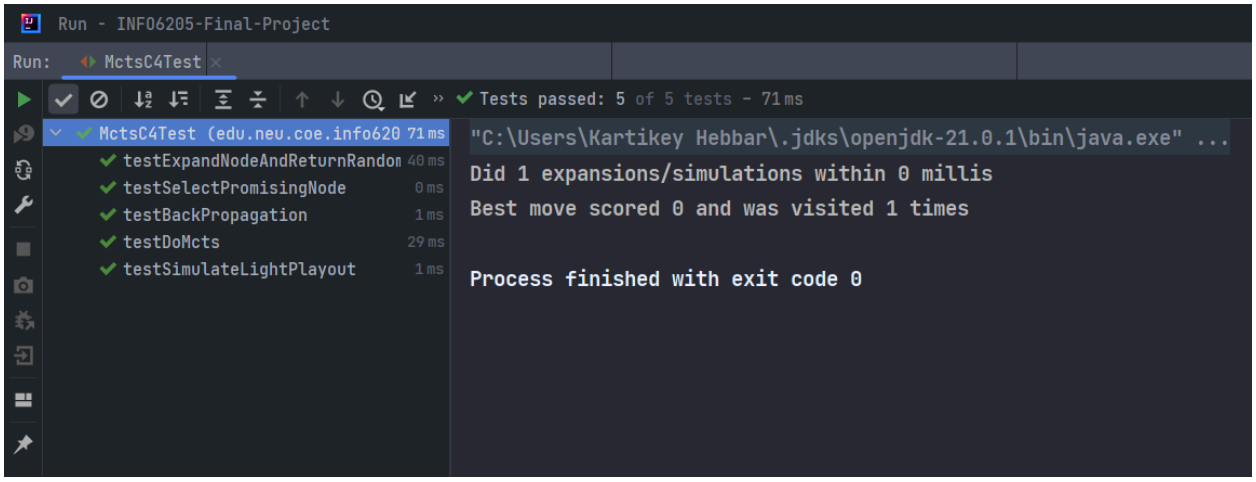

Run - INFO6205-Final-Project
Run: Main
Player move
23
			x			
		o	o	x		
		x	x	o	x	
		o	x	o	o	
0	1	2	3	4	5	6
Computer move
Doing MCTS.
Did 4000 expansions/simulations within 14 millis
Best move scored 2909 and was visited 2909 times
MCTS done.
winning player is: Computer
			x			
		o	o	x		
		x	x	o	x	
		o	x	o	o	x
0	1	2	3	4	5	6

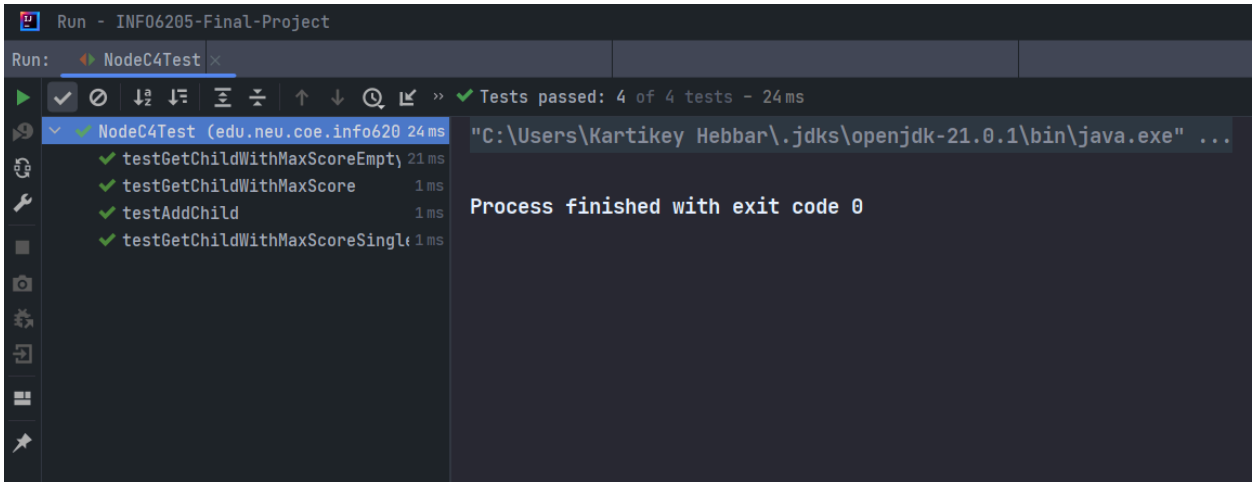

			x			
		o	o	x		
		x	x	o	x	
		o	x	o	o	x
0	1	2	3	4	5	6
All MCTS operations took: 828 milliseconds

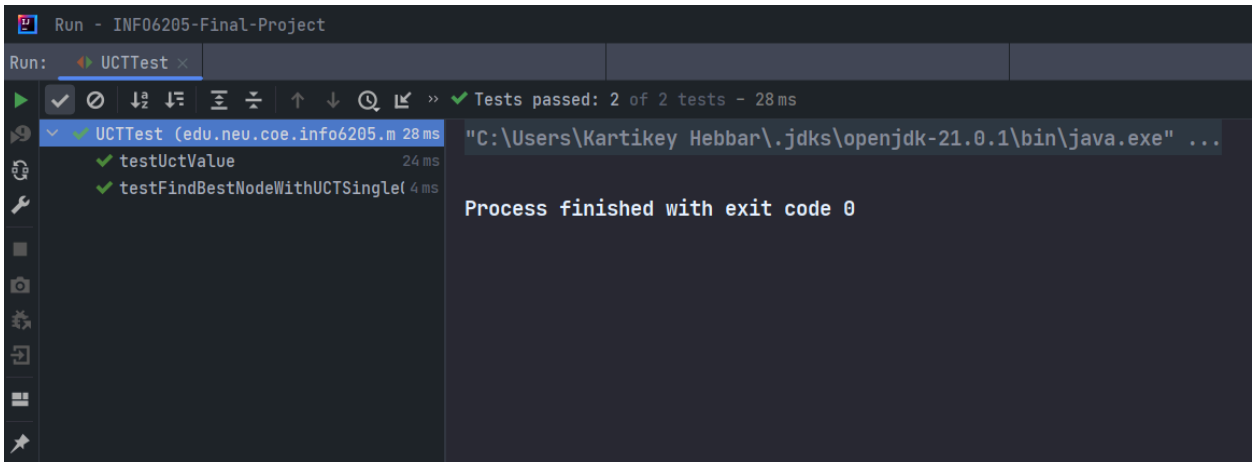


Results: The above graph shows the time taken for MCTS to consider all moves to win a Connect-4 game vs the number of iterations, Please note that all times shown are in milliseconds(ms)

UNIT TEST SNAPSHOTS







REFERENCES

1. https://www.researchgate.net/figure/Outline-of-Monte-Carlo-Tree-Search-adapted-from-Chaslot-et-al-39_fig2_224160689
2. https://en.wikipedia.org/wiki/Monte_Carlo_tree_search
3. https://www.flaticon.com/free-icon/tic-tac-toe_10199746
4. https://en.wikipedia.org/wiki/Tic-tac-toe
5. https://www.baeldung.com/java-monte-carlo-tree-search
6. https://www.rd.com/article/how-to-win-connect-4/
7. https://en.wikipedia.org/wiki/Connect_Four
8. https://www.harrycodes.com/blog/monte-carlo-tree-search