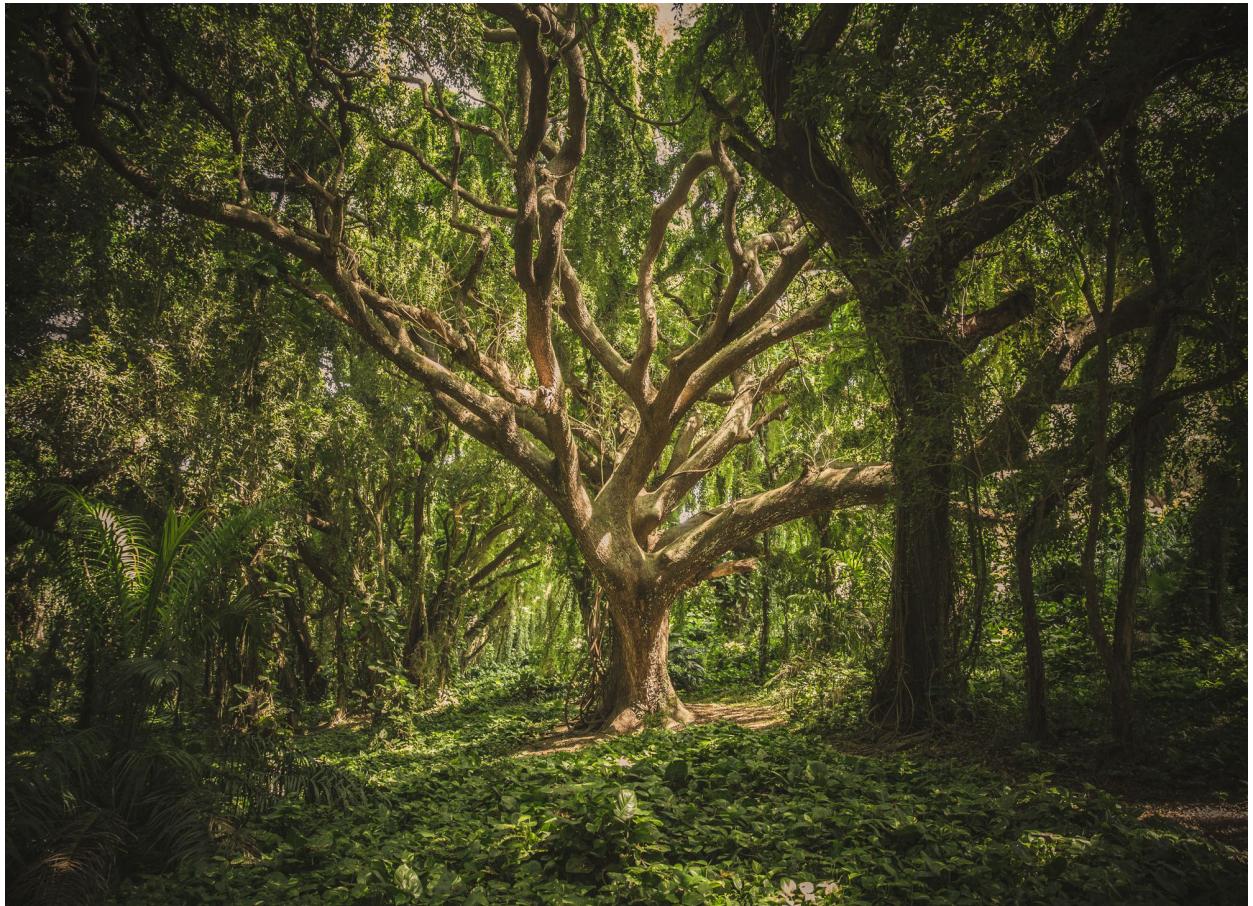


# **Tic-Tac-Toe & Connect Four Game**

## **FINAL PROJECT**

**Rahul Pothirendi, Rohit Varma Mudundi**  
**Software Engineering Systems, Northeastern University**  
[pothirendi.r@northeastern.edu](mailto:pothirendi.r@northeastern.edu), [mudundi.r@northeastern.edu](mailto:mudundi.r@northeastern.edu)

**Abstract** – This project focuses on creating a Connect Four game using Monte Carlo Tree Search (MCTS). The report discusses how the idea for the game originated and the challenges encountered during development. It explains the importance of MCTS in the project and how Tic-Tac-Toe served as a model for building the Connect Four game.



## PROBLEM DESCRIPTION

In this project, we're tasked with developing our own game using Monte Carlo Tree Search (MCTS), an algorithm for decision-making in certain types of games. MCTS involves four key steps:

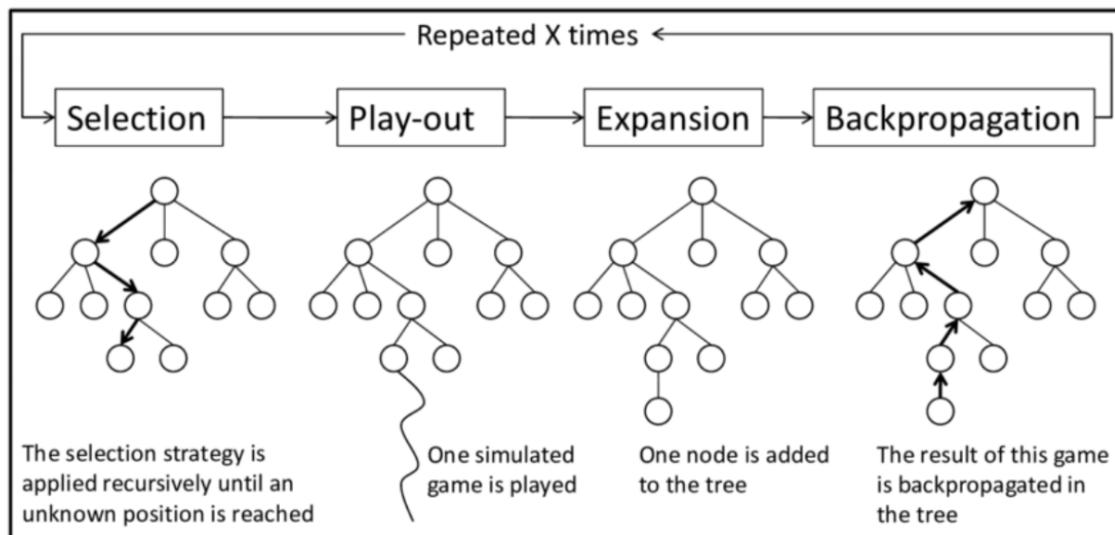
Selection, Expansion, Simulation, and Backpropagation.

**Selection** begins at the root node and navigates through the tree based on a selection policy until reaching an unexplored node.

**Expansion** involves adding child nodes to expand the tree.

**Simulation** entails playing out the game from the current node to the end, recording results for all possible outcomes, and selecting the best solution.

**Backpropagation** involves updating the information stored in the nodes traversed to reach the terminal node. Understanding and implementing these four steps are essential for the search algorithm to function effectively.



## ANALYSIS

Before starting our main project, we practiced with Tic-Tac-Toe to understand how Monte Carlo Tree Search (MCTS) works in a game. In Tic-Tac-Toe, we focused on managing the game's state using a 3x3 grid. We defined empty spaces as -1, 'O' as 0, and 'X' as 1. We wrote code to generate possible moves and check for winning combinations in rows, columns, and diagonals.

The screenshot shows a Java development environment with the following details:

- Project Structure:** The project is named "info6205-project" and contains files: TicTacToe.java, State.java, TicTacToeTest.java (selected), TicTacToeNodeTest.java, MCTSTest.java, PositionTest.java, MCTS.java, and TicTacToeNode.java.
- TicTacToeTest.java Content:**

```
1 package edu.neu.coe.info6205.mcts.tictactoe;
2
3 > import ...
4
5 & pothirendirahul
6 public class TicTacToeTest {
7
8     /**
9      *
10     */
11    & pothirendirahul
12    @Test
13    public void runGame() {
14        long seed = 0L;
15        TicTacToe target = new TicTacToe(seed); // games run here will all be deterministic.
16        State<TicTacToe> state = target.runGame();
17        Optional<Integer> winner = state.winner();
18        if (winner.isPresent()) assertEquals(Integer.valueOf(TicTacToe.X), winner.get());
19        else fail("no winner");
20    }
21 }
```
- Run Tab:** The "Run" tab is selected, showing the test results for "TicTacToeTest".
- Test Results:**
  - TicTacToeTest (edu.neu.coe)**: Tests passed: 1 of 1 test – 7ms
  - runGame**: 7ms /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...  
Process finished with exit code 0

Unit test for TicTacToe

The screenshot shows an IDE interface with the following details:

- Project View:** Shows a tree structure of files and folders. Under the 'mcts' folder, there is a 'ConnectFour' package containing four test classes: ConnectFourMoveTest, ConnectFourNodeTest, ConnectFourPositionTest, and ConnectFourTest.
- Code Editor:** The right pane displays the code for the `ConnectFourTest.java` file. The code includes a `runGame()` method and a `testPlayerAlternation()` test method.
- Run Tab:** The 'Run' tab is active, showing the results of a recent run. It lists 12 tests with their execution times:
  - testPlayerAlternation (10ms)
  - testMoves (0ms)
  - testNext (1ms)
  - testMainMethod (127ms)
  - testTerminalState (80ms)
  - testConnectFourPosition (1ms)
  - runGame (61ms)
  - testIsTerminal (0ms)
  - testPlayer (0ms)
  - testRandom (0ms)
  - testWinner (1ms)
  - testMoveIterator (1ms)
- Output Panel:** Below the run results, it says "Process finished with exit code 0".
- Status Bar:** At the bottom, it shows the path "info6205-project > src > test > java > edu > neu > coe > info6205 > mcts > ConnectFour > ConnectFourTest", and the status "47:1 LF UTF-8 4 spaces".

## ConnectFourTest

### ConnectFour Unit Tests Summary

`testMainMethod`: Tests the main method of the ConnectFour class. It checks the output to ensure it contains either "ConnectFour: Winner is:" or "ConnectFour: Draw".

`runGame`: Tests the runGame method of the ConnectFour class. This test runs a deterministic game and verifies if the winner is player X.

`testPlayerAlternation`: Verifies if player alternation is correctly implemented. It checks if the player changes after making a move.

`testTerminalState`: Checks if the game reaches a terminal state after running.

`testConnectFourPosition`: Tests the ConnectFourPosition class. It ensures correct initialization and checks the grid for expected values.

`testIsTerminal`: Validates if the game state is correctly identified as terminal.

`testPlayer`: Verifies if the current player is correctly identified.

`testWinner`: Checks if there is no winner at the start of the game.

`testRandom`: Ensures proper initialization of a random number generator.

`testMoves`: Checks if there are available moves for the current player.

`testNext`: Tests if the next method correctly generates the next state after making a move.

`testMoveIterator`: Validates if the move iterator is properly initialized.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the "mcts" package, including files like "ConnectFourMoveTest", "ConnectFourNodeTest", "ConnectFourPositionTest", and "MCTSTest".
- Code Editor:** The main editor window displays the `ConnectFourPositionTest.java` file. The code defines a test class `ConnectFourPositionTest` with several test methods using `@Test` annotations.
- Run Tab:** The bottom-left tab bar is set to "Run".
- Output Window:** The bottom-right panel shows the test results:
  - Tests passed:** 8 of 8 tests - 18 ms
  - Test details:
    - testMoveColumnFull (12 ms)
    - testMoves (0 ms)
    - testParseCell (0 ms)
    - testFull (1 ms)
    - testMove (0 ms)
    - testReflect (3 ms)
    - testWinner (1 ms)
    - testParsePosition (1 ms)
  - Message: "Process finished with exit code 0"
- Status Bar:** The bottom status bar shows the path "info6205-project > src > test > java > edu > neu > coe > info6205 > mcts > ConnectFour > ConnectFourPositionTest", the time "13:14", and file encoding "UTF-8".

## ConnectFour Position Test

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the "mcts" package, including files like "ConnectFourMoveTest", "ConnectFourNodeTest", "ConnectFourPositionTest", and "MCTSTest".
- Code Editor:** The main editor window displays the `ConnectFourNodeTest.java` file. The code defines a test class `ConnectFourNodeTest` with several test methods using `@Test` annotations.
- Run Tab:** The bottom-left tab bar is set to "Run".
- Output Window:** The bottom-right panel shows the test results:
  - Tests passed:** 6 of 6 tests - 11 ms
  - Test details:
    - addChild (8 ms)
    - state (1 ms)
    - white (1 ms)
    - backPropagate (1 ms)
    - children (0 ms)
    - winsAndPlayouts (0 ms)
  - Message: "Process finished with exit code 0"
- Status Bar:** The bottom status bar shows the path "info6205-project > src > test > java > edu > neu > coe > info6205 > mcts > ConnectFour > ConnectFourNodeTest", the time "7:14", and file encoding "UTF-8".

## Connect Four Node Test

The screenshot shows an IDE interface with the PositionTest.java file open. The code contains several test methods for the Position class, including testFull(), testRender(), and testToString(). The run results show 18 tests passed in 16 ms.

```
134     @Test
135     public void testFull() {
136         assertFalse(Position.parsePosition("X . 0\nX 0 .\nX . 0", last).isFull());
137         assertTrue(Position.parsePosition("X X 0\nX 0 0\nX X 0", last).isFull());
138     }
139
140     @Test
141     public void testRender() {
142         String grid = "X . .\n. 0 .\n. . X";
143         Position target = Position.parsePosition(grid, last);
144         assertEquals(grid, target.render());
145     }
146
147     @Test
148     public void testToString() {
149         Position target = Position.parsePosition("X . .\n. 0 .\n. . X", last);
150         assertEquals("1,-1,-1\n-1,0,-1\n-1,-1,1", target.toString());
151     }
152
153
Run PositionTest
T PositionTest (edu.neu.co 16 ms)
    ✓ Tests passed: 18 of 18 tests - 16 ms
    /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...
Process finished with exit code 0
```

## Unit test for PositionTest

## Unit test for TicTacToe Node

The screenshot shows an IDE interface with the TicTacToeNodeTest.java file open. The code contains tests for the TicTacToeNode class, including winsAndLayouts() and state(). The run results show 6 tests passed in 12 ms.

```
7
8     public class TicTacToeNodeTest {
9
10         @Test
11         public void winsAndLayouts() {
12             TicTacToe.TicTacToeState state = new TicTacToe().new TicTacToeState(Position.parsePosition("X . 0\nX 0 .\nX . 0", TicTacToe.X));
13             TicTacToeNode node = new TicTacToeNode(state);
14             assertTrue(node.isLeaf());
15             assertEquals(expected: 2, node.wins());
16             assertEquals(expected: 1, node.layouts());
17         }
18
19         @Test
20         public void state() {
21             TicTacToe.TicTacToeState state = new TicTacToe().new TicTacToeState();
22             TicTacToeNode node = new TicTacToeNode(state);
23             assertEquals(state, node.state());
24         }
25
Run TicTacToeNodeTest
T TicTacToeNodeTest (edu.n 12 ms)
    ✓ Tests passed: 6 of 6 tests - 12 ms
    /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...
Process finished with exit code 0
```

## IMPLEMENTATION

As we brainstormed our game concept, we considered developing a Connect Four game. The game would start with a grid where players can drop their tokens. The grid would have multiple columns and rows. Players take turns dropping their tokens into a column, and the tokens stack up vertically. The objective is to connect four of their tokens vertically, horizontally, or diagonally before their opponent does. For example, a player might choose column 3 and drop their token into it, causing the token to stack on top of any existing tokens in the column.

## IMPLEMENTATION OF TICTACTOE.JAVA

```
info6205-project main
```

```
public static void main(String[] args) {
    int[] totalGames = new int[]{200, 400, 800, 1600, 3200, 6400};

    for (int n : totalGames) {
        runGames(n);
    }
}

1 usage  ^ pothirendhirahul
public static void runGames(int totalGames) {
    int playerXCount = 0;
```

The screenshot shows the Java code for the `TicTacToe` class. The `main` method initializes an array of total games (200, 400, 800, 1600, 3200, 6400) and iterates through them, calling the `runGames` method for each. The `runGames` method takes an integer parameter for the total number of games. The code then prints the results to the terminal, showing the total number of games, Player X (X) Wins, Player O (O) Wins, and Draws for each group of games. The terminal output is as follows:

```
Total Games Player X (X) Wins Player O (O) Wins Draws
200 117 83 0
-----
Total Games Player X (X) Wins Player O (O) Wins Draws
400 329 52 19
-----
Total Games Player X (X) Wins Player O (O) Wins Draws
800 526 274 0
-----
Total Games Player X (X) Wins Player O (O) Wins Draws
1600 826 461 313
-----
Total Games Player X (X) Wins Player O (O) Wins Draws
3200 1799 963 498
-----
Total Games Player X (X) Wins Player O (O) Wins Draws
6400 3946 1326 1128
-----
Process finished with exit code 0
```

At the bottom of the terminal window, there is a status bar with the following information: info6205-project > src > main > java > edu > neu > coe > info6205 > mcts > tictactoe > TicTacToe > main. The status bar also shows the time as 21:10, LF, UTF-8, 4 spaces, and a small icon.

```
1 package edu.neu.coe.info6205.mcts.tictactoe;
2
3 import edu.neu.coe.info6205.mcts.core.Node;
4
5 /**
6 * Class to represent a Monte Carlo Tree Search for Tic Tac Toe.
7 */
8
9 @pothirendhirahul
10 public class MCTS {
11
12     @pothirendhirahul
13     public static void main(String[] args) {
14         MCTS mcts = new MCTS(new TicTacToeNode(new TicTacToe().new TicTacToeState()));
15         Node<TicTacToe> root = mcts.root;
16
17         int iterations = 1200;
18         mcts.runMCTS(iterations, root);
19     }
20
21     @pothirendhirahul
22     public MCTS(Node<TicTacToe> root) {
23         this.root = root;
24     }
25
26     @pothirendhirahul
27     public void runMCTS(int iterations, Node<TicTacToe> root) {
28         for (int i = 0; i < iterations; i++) {
29             TicTacToeNode promisingNode = selectPromisingNode((TicTacToeNode) root);
30
31             @pothirendhirahul
32             promisingNode.expand();
33
34             @pothirendhirahul
35             TicTacToeNode visitNode = visitNode(promisingNode);
36
37             @pothirendhirahul
38             updateNode(visitNode);
39
40             @pothirendhirahul
41             backPropagate(visitNode);
42         }
43     }
44
45     private final Node<TicTacToe> root;
46
47     @pothirendhirahul
48     TicTacToeNode selectPromisingNode(TicTacToeNode root) {
49         return null;
50     }
51
52     @pothirendhirahul
53     TicTacToeNode visitNode(TicTacToeNode node) {
54         return null;
55     }
56
57     @pothirendhirahul
58     void updateNode(TicTacToeNode node) {
59     }
60
61     @pothirendhirahul
62     void backPropagate(TicTacToeNode node) {
63     }
64 }
```

## MCTS TicTacToe

```
. X . . X . 0
. X . 0 X . X
. 0 . X X 0 0
0 0 . X 0 0 X
X X 0 0 X X 0
0 0 0 X X X 0
```

```
. X . . X . 0
. X . 0 X . X
. 0 . X X 0 0
0 0 . X 0 0 X
X X 0 0 X X 0
0 0 0 X X X 0
```

ConnectFour: Winner is: 0 (Random)

ConnectFour: The Winner is: 0

Process finished with exit code 0

```
* @param seed The seed for the random number generator.
*/
2 usages new*
public ConnectFour(long seed) {
    this(new Random(seed));
}

/**
 * Creates the starting position for the Connect Four game.
 * @return The starting position.
 */
1 usage new*
static ConnectFourPosition startingPosition() {
    return ConnectFourPosition.parsePosition(
        grid: ". . . . . \n" +
            ". . . . . ", BLANK);
}

/**
 * Starts a new Connect Four game.
 * @return The initial state of the game.
 */
new*
public State<ConnectFour> start() {
    return new ConnectFourState();
}

/**
 * Specifies which player makes the first move.
 * @return The player symbol for the opener.
*/
```

## EVALUATION

With these implementations in place, we tested our game to ensure its functionality. The screenshot below depicts the appearance of the Connect Four game, with the grid correctly generated and the game logic functioning as intended. Additionally, other sections of the program include unit tests to verify the proper execution of methods.

Total Games	Player X (X) Wins	Player O (O) Wins	Draws
200	104	78	18
400	234	166	0
800	381	387	32
1600	703	660	237
3200	1850	988	362
6400	3152	2122	1126

## CONCLUSIONS

In this project, we recognize MCTS as a search algorithm utilized in decision-making games, enabling complex decision-making through Selection, Expansion, Simulation, and Backpropagation. Initially, we grasped the concept of MCTS by engaging with a sample Tic-Tac-Toe game. Subsequently, we delved further into MCTS by crafting our own Connect Four game.

## REFERENCES

## **FOR TICTACTOE**

<https://www.baeldung.com/java-monte-carlo-tree-search>

<https://martin-ueding.de/posts/tic-tac-toe-with-monte-carlo-tree-search/>

## **FOR MCTS CONNECT FOUR**

<https://ai.stackexchange.com/questions/21019/should-monte-carlo-tree-search-be-able-to-consistently-beat-me-in-the-connect-four-game>

<https://link.springer.com/article/10.1007/s10462-022-10228-y>